

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

```
1 #include <iostream>
2 #include <cstring>
3 using namespace std;
4
5 // Deklarasi Struct untuk mahasiswa
6 struct mahasiswa
7 {
8     char nama[30];
9     char nim[10];
10 };
11
12 // Deklarasi Struct Node
13 struct Node
14 {
15     mahasiswa data;
16     Node *next;
17 };
18
19 Node *head;
20 Node *tail;
21
22 // Inisialisasi list
23 void init()
24 {
25     head = nullptr;
26     tail = nullptr;
27 }
28
29 // Pengecekan apakah list kosong
30 bool isEmpty()
31 {
32     return head == nullptr;
33 }
34
35 // Tambah Depan
36 void insertDepan(const mahasiswa &data)
37 {
38     Node *baru = new Node;
39     baru->data = data;
40     baru->next = nullptr;
41     if (isEmpty())
42     {
43         head = tail = baru;
44     }
45     else
46     {
47         baru->next = head;
48         head = baru;
49     }
50 }
51
52 // Tambah Belakang
53 void insertBelakang(const mahasiswa &data)
54 {
55     Node *baru = new Node;
56     baru->data = data;
57     baru->next = nullptr;
58     if (isEmpty())
59     {
60         head = tail = baru;
61     }
62     else
63     {
64         tail->next = baru;
65         tail = baru;
66     }
67 }
68
69 // Hitung Jumlah List
70 int hitungList()
71 {
72     Node *current = head;
73     int jumlah = 0;
74     while (current != nullptr)
75     {
76         jumlah++;
77         current = current->next;
78     }
79     return jumlah;
80 }
81
82 // Hapus Depan
83 void hapusDepan()
84 {
85     if (isEmpty())
86     {
87         Node *hapus = head;
88         head = head->next;
89         delete hapus;
90         if (head == nullptr)
91         {
92             tail = nullptr; // Jika list menjadi kosong
93         }
94     }
95     else
96     {
97         cout << "List kosong!" << endl;
98     }
99 }
100
```

```

1 // Hapus Belakang
2 void hapusBelakang()
3 {
4     if (isEmpty())
5     {
6         if (head == tail)
7         {
8             delete head;
9             head = tail = nullptr; // List menjadi kosong
10        }
11        else
12        {
13            Node *bantu = head;
14            while (bantu->next != tail)
15            {
16                bantu = bantu->next;
17            }
18            delete tail;
19            tail = bantu;
20            tail->next = nullptr;
21        }
22    }
23    else
24    {
25        cout << "list kosong!" << endl;
26    }
27 }
28
29 // Tampilkan List
30 void tampil()
31 {
32     Node *current = head;
33     if (isEmpty())
34     {
35         while (current != nullptr)
36         {
37             cout << "Nama: " << current->data.nama << ", NIM: " << current->data.nim << endl;
38             current = current->next;
39         }
40     }
41     else
42     {
43         cout << "list masih kosong!" << endl;
44     }
45     cout << endl;
46 }
47
48 // Hapus List
49 void clearList()
50 {
51     Node *current = head;
52     while (current != nullptr)
53     {
54         Node *hapus = current;
55         current = current->next;
56         delete hapus;
57     }
58     head = tail = nullptr;
59     cout << "list berhasil terhapus!" << endl;
60 }
61
62 // Main function
63 int main()
64 {
65     init();
66
67     // Contoh data mahasiswa
68     mahasiswa m1 = {"Alice", "123456"};
69     mahasiswa m2 = {"Bob", "654321"};
70     mahasiswa m3 = {"Charlie", "112233"};
71
72     // Menambahkan mahasiswa ke dalam list
73     insertDepan(m1);
74     tampil();
75     insertBelakang(m2);
76     tampil();
77     insertDepan(m3);
78     tampil();
79
80     // Menghapus elemen dari list
81     hapusDepan();
82     tampil();
83     hapusBelakang();
84     tampil();
85
86     // Menghapus seluruh list
87     clearList();
88
89     return 0;
90 }

```

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>

```

Guided 2

```

1 #include <iostream>
2 using namespace std;
3
4 // Definisi struktur untuk elemen list
5 struct Node
6 {
7     int data; // Menyimpan nilai elemen
8     Node *next; // Pointer ke elemen berikutnya
9 };
10
11 // Fungsi untuk mengalokasikan memori untuk node baru
12 Node *alokasi(int value)
13 {
14     Node *newNode = new Node; // Alokasi memori untuk elemen baru
15     if (newNode != nullptr)
16     {
17         // Jika alokasi berhasil
18         newNode->data = value; // Mengisi data node
19         newNode->next = nullptr; // Set next ke nullptr
20     }
21     return newNode; // Mengembalikan pointer node baru
22 }
23
24 // Fungsi untuk dealokasi memori node
25 void dealokasi(Node *node)
26 {
27     delete node; // Mengembalikan memori yang digunakan oleh node
28 }
29
30 // Mengecek apakah list kosong
31 bool isEmpty(Node *head)
32 {
33     return head == nullptr; // List kosong jika head adalah nullptr
34 }
35
36 // Menambahkan elemen di awal list
37 void insertFirst(Node *head, int value)
38 {
39     Node *newNode = alokasi(value); // Alokasi memori untuk elemen baru
40     if (newNode != nullptr)
41     {
42         newNode->next = head; // Menghubungkan elemen baru ke elemen pertama
43         head = newNode; // Menetapkan elemen baru sebagai elemen pertama
44     }
45 }
46
47 // Menambahkan elemen di akhir list
48 void insertLast(Node *head, int value)
49 {
50     Node *newNode = alokasi(value); // Alokasi memori untuk elemen baru
51     if (newNode != nullptr)
52     {
53         if (isEmpty(head))
54         {
55             // Jika list kosong
56             head = newNode; // Elemen baru menjadi elemen pertama
57         }
58         else
59         {
60             Node *temp = head;
61             while (temp->next != nullptr)
62             {
63                 // Mencari elemen terakhir
64                 temp = temp->next;
65             }
66             temp->next = newNode; // Menambahkan elemen baru di akhir list
67         }
68     }
69 }
70
71 // Menampilkan semua elemen dalam list
72 void printList(Node *head)
73 {
74     if (isEmpty(head))
75     {
76         cout << "List kosong!" << endl;
77     }
78     else
79     {
80         Node *temp = head;
81         while (temp != nullptr)
82         {
83             // Selama belum mencapai akhir list
84             cout << temp->data << " "; // Menampilkan data elemen
85             temp = temp->next; // Melanjutkan ke elemen berikutnya
86         }
87         cout << endl;
88     }
89 }
90
91 // Menghitung jumlah elemen dalam list
92 int countElements(Node *head)
93 {
94     int count = 0;
95     Node *temp = head;
96     while (temp != nullptr)
97     {
98         // Menambah jumlah elemen
99         count++;
100         temp = temp->next; // Melanjutkan ke elemen berikutnya
101     }
102     return count; // Mengembalikan jumlah elemen
103 }
104
105 // Menghapus semua elemen dalam list dan dealokasi memori
106 void clearList(Node *head)
107 {
108     while (head != nullptr)
109     {
110         Node *temp = head; // Simpan pointer ke node saat ini
111         head = head->next; // Pindahkan ke node berikutnya
112         dealokasi(temp); // Dealokasi node
113     }
114 }
115
116 int main()
117 {
118     Node *head = nullptr; // Membuat list kosong
119
120     // Menambahkan elemen ke dalam list
121     insertFirst(head, 10); // Menambahkan elemen 10 di awal list
122     insertLast(head, 20); // Menambahkan elemen 20 di akhir list
123     insertLast(head, 30); // Menambahkan elemen 30 di akhir list
124
125     // Menampilkan isi list
126     cout << "Isi list: ";
127     printList(head);
128
129     // Menampilkan jumlah elemen
130     cout << "Jumlah elemen: " << countElements(head) << endl;
131
132     // Menghapus semua elemen dalam list
133     clearList(head);
134
135     // Menampilkan isi list setelah penghapusan
136     cout << "Isi list setelah penghapusan: ";
137     printList(head);
138
139     return 0;
140 }

```

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>
```

4. Unguided

1.

```
1  #include <iostream>
2  using namespace std;
3
4  // Node structure
5  struct Node
6  {
7      int data;
8      Node *next;
9  };
10
11 // Head pointer to the linked list
12 Node *head = nullptr;
13
14 // Function to insert a node at the front
15 void insertDepan(int nilai)
16 {
17     Node *newNode = new Node();
18     newNode->data = nilai;
19     newNode->next = head;
20     head = newNode;
21 }
22
23 // Function to insert a node at the back
24 void insertBelakang(int nilai)
25 {
26     Node *newNode = new Node();
27     newNode->data = nilai;
28     newNode->next = nullptr;
29
30     if (head == nullptr)
31     {
32         head = newNode;
33     }
34     else
35     {
36         Node *temp = head;
37         while (temp->next != nullptr)
38         {
39             temp = temp->next;
40         }
41         temp->next = newNode;
42     }
43 }
44
45 // Function to print the entire linked list
46 void cetakLinkedList()
47 {
48     Node *temp = head;
49     if (temp == nullptr)
50     {
51         cout << "Linked list kosong" << endl;
52     }
53     else
54     {
55         while (temp != nullptr)
56         {
57             cout << temp->data;
58             if (temp->next != nullptr)
59             {
60                 cout << " -> ";
61             }
62             temp = temp->next;
63         }
64         cout << endl;
65     }
66 }
67
68 int main()
69 {
70     // Contoh operasi
71     insertDepan(10); // Tambah node di depan (nilai: 10)
72     insertBelakang(20); // Tambah node di belakang (nilai: 20)
73     insertDepan(5); // Tambah node di depan (nilai: 5)
74
75     // Cetak isi linked list
76     cetakLinkedList(); // Output: 5 -> 10 -> 20
77
78     return 0;
79 }
```

Output

```
for-riz4ruggw.yar --pid=MI  
5 -> 10 -> 20  
PS D:\guided modul 4>
```


2.

```

1 #include <iostream>
2 using namespace std;
3
4 // Node structure
5 struct Node
6 {
7     int data;
8     Node *next;
9 };
10
11 // Head pointer to the linked list
12 Node *head = nullptr;
13
14 // Function to insert a node at the front
15 void insertDepan(int nilai)
16 {
17     Node *newNode = new Node();
18     newNode->data = nilai;
19     newNode->next = head;
20     head = newNode;
21 }
22
23 // Function to insert a node at the back
24 void insertBelakang(int nilai)
25 {
26     Node *newNode = new Node();
27     newNode->data = nilai;
28     newNode->next = nullptr;
29
30     if (head == nullptr)
31     {
32         head = newNode;
33     }
34     else
35     {
36         Node *temp = head;
37         while (temp->next != nullptr)
38         {
39             temp = temp->next;
40         }
41         temp->next = newNode;
42     }
43 }
44
45 // Function to delete a node with a specific value
46 void hapusNode(int nilai)
47 {
48     if (head == nullptr)
49     {
50         cout << "Linked list kosong, tidak ada node untuk dihapus" << endl;
51         return;
52     }
53
54     // Jika node yang harus dihapus adalah head
55     if (head->data == nilai)
56     {
57         Node *temp = head;
58         head = head->next;
59         delete temp;
60         cout << "Node dengan nilai " << nilai << " berhasil dihapus" << endl;
61         return;
62     }
63
64     // Mencari node dengan nilai tertentu
65     Node *temp = head;
66     Node *prev = nullptr;
67     while (temp != nullptr && temp->data != nilai)
68     {
69         prev = temp;
70         temp = temp->next;
71     }
72
73     // Jika node tidak ditemukan
74     if (temp == nullptr)
75     {
76         cout << "Node dengan nilai " << nilai << " tidak ditemukan" << endl;
77         return;
78     }
79
80     // Hapus node
81     prev->next = temp->next;
82     delete temp;
83     cout << "Node dengan nilai " << nilai << " berhasil dihapus" << endl;
84 }
85
86 // Function to print the entire linked list
87 void cetakLinkedList()
88 {
89     Node *temp = head;
90     if (temp == nullptr)
91     {
92         cout << "Linked list kosong" << endl;
93     }
94     else
95     {
96         while (temp != nullptr)
97         {
98             cout << temp->data;
99             if (temp->next != nullptr)
100             {
101                 cout << " -> ";
102             }
103             temp = temp->next;
104         }
105         cout << endl;
106     }
107 }
108
109 int main()
110 {
111     // Contoh operasi
112     insertDepan(10); // Tambah node di depan (nilai: 10)
113     insertBelakang(20); // Tambah node di belakang (nilai: 20)
114     insertDepan(5); // Tambah node di depan (nilai: 5)
115
116     // Cetak isi linked list sebelum penghapusan
117     cetakLinkedList(); // Output: 5 -> 10 -> 20
118
119     // Hapus node dengan nilai tertentu
120     hapusNode(10); // Hapus node dengan nilai 10
121
122     // Cetak isi linked list setelah penghapusan
123     cetakLinkedList(); // Output: 5 -> 20
124
125     return 0;
126 }

```

Output

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

3.

```

1  #include <iostream>
2  using namespace std;
3
4  // Node structure
5  struct Node
6  {
7      int data;
8      Node *next;
9  };
10
11 // Head pointer to the linked list
12 Node *head = nullptr;
13
14 // Function to insert a node at the front
15 void insertDepan(int nilai)
16 {
17     Node *newNode = new Node();
18     newNode->data = nilai;
19     newNode->next = head;
20     head = newNode;
21 }
22
23 // Function to insert a node at the back
24 void insertBelakang(int nilai)
25 {
26     Node *newNode = new Node();
27     newNode->data = nilai;
28     newNode->next = nullptr;
29
30     if (head == nullptr)
31     {
32         head = newNode;
33     }
34     else
35     {
36         Node *temp = head;
37         while (temp->next != nullptr)
38         {
39             temp = temp->next;
40         }
41         temp->next = newNode;
42     }
43 }
44
45 // Function to search for a node with a specific value
46 bool cariNode(int nilai)
47 {
48     Node *temp = head;
49     while (temp != nullptr)
50     {
51         if (temp->data == nilai)
52         {
53             return true;
54         }
55         temp = temp->next;
56     }
57     return false;
58 }
59
60 // Function to count the length of the linked list
61 int hitungPanjangLinkedList()
62 {
63     Node *temp = head;
64     int count = 0;
65     while (temp != nullptr)
66     {
67         count++;
68         temp = temp->next;
69     }
70     return count;
71 }
72
73 // Function to print the entire linked list
74 void cetakLinkedList()
75 {
76     Node *temp = head;
77     if (temp == nullptr)
78     {
79         cout << "Linked list kosong" << endl;
80     }
81     else
82     {
83         while (temp != nullptr)
84         {
85             cout << temp->data;
86             if (temp->next != nullptr)
87             {
88                 cout << " -> ";
89             }
90             temp = temp->next;
91         }
92         cout << endl;
93     }
94 }
95
96 int main()
97 {
98     // Contoh operasi
99     insertDepan(10); // Tambah node di depan (nilai: 10)
100    insertBelakang(20); // Tambah node di belakang (nilai: 20)
101    insertDepan(5); // Tambah node di depan (nilai: 5)
102
103    // Cetak isi linked list
104    cetakLinkedList(); // Output: 5 -> 10 -> 20
105
106    // Mencari node dengan nilai tertentu
107    int nilaiDicari = 20;
108    if (cariNode(nilaiDicari))
109    {
110        cout << "Node dengan nilai " << nilaiDicari << " ditemukan." << endl;
111    }
112    else
113    {
114        cout << "Node dengan nilai " << nilaiDicari << " tidak ditemukan." << endl;
115    }
116
117    // Hitung panjang linked list
118    int panjang = hitungPanjangLinkedList();
119    cout << "Panjang linked list: " << panjang << endl;
120
121    return 0;
122 }

```

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.