

LAPORAN PRAKTIKUM  
Modul 04  
“SINGLE LINKED LIST”



Disusun Oleh:  
Faishal Arif Setiawan 2311104066  
Kelas:  
**SE 07 02**  
Dosen :  
WAHYU ANDI SAPUTRA, S.PD.,M.ENG

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

**2.1** 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*.

Setiap elemen dalam linked list disebut sebagai node, dan setiap node terdiri dari dua komponen:

- Data: Nilai yang disimpan di dalam node.
- Pointer (next): Penunjuk ke node berikutnya dalam urutan linked list.

### 2.2. Single linked list

adalah struktur data linear di mana setiap elemen (node) hanya memiliki satu penunjuk (pointer) yang mengarah ke node berikutnya. Elemen pertama dalam linked list disebut head, dan node terakhir memiliki penunjuk yang bernilai null, yang menunjukkan bahwa node tersebut merupakan elemen terakhir dalam linked list.

Sifat dari *Single Linked list*:

1. Hanya memerlukan satu buah *pointer*.
2. *Node* akhir menunjuk ke Nil kecuali untuk *list circular*.
3. Hanya dapat melakukan pembacaan maju.
4. Pencarian sequensial dilakukan jika data tidak terurut.
5. Lebih mudah ketika melakukan penyisipan atau penghapusan di tengah *list*

## 3.GUIDED



Program C++ di atas adalah implementasi linked list untuk menyimpan data mahasiswa menggunakan single linked list. Program ini menyediakan beberapa operasi dasar, seperti menambahkan node di depan dan belakang, menghapus node di depan dan belakang, menampilkan isi list, serta menghapus seluruh isi linked list.

Struct mahasiswa: Menyimpan data mahasiswa berupa nama dan nim

Struct Node: Merepresentasikan node pada linked list, yang terdiri dari data mahasiswa dan pointer next untuk menunjuk ke node berikutnya.

init(): Inisialisasi list dengan mengatur head dan tail menjadi nullptr, menandakan bahwa list kosong.

isEmpty(): Memeriksa apakah linked list kosong. Mengembalikan true jika list kosong.

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!
```

```

1 // Fungsi untuk menghapus semua elemen dari list
2 void hapusSemuaElemen(List)
3 {
4     // Iterasi untuk menghapus semua elemen dari list
5     if (head == NULL) // Jika list kosong
6     {
7         return;
8     }
9     // Fungsi untuk menghapus semua elemen dari list
10    Node* temp = head; // Pointer ke elemen pertama
11    while (temp != NULL) // Selama list tidak kosong
12    {
13        // Simpan alamat elemen berikutnya
14        Node* next = temp->next;
15        // Hapus elemen saat ini
16        delete temp;
17        temp = next;
18    }
19    // Set head ke NULL
20    head = NULL;
21 }
22
23 // Fungsi untuk mencari elemen dalam list
24 int cariElemen(List, int value)
25 {
26     // Iterasi untuk mencari elemen
27     Node* temp = head; // Pointer ke elemen pertama
28     while (temp != NULL) // Selama list tidak kosong
29     {
30         // Cek apakah nilai sama
31         if (temp->data == value)
32         {
33             return 1; // Elemen ditemukan
34         }
35         temp = temp->next;
36     }
37     return 0; // Elemen tidak ditemukan
38 }
39
40 // Fungsi untuk menambahkan elemen di awal list
41 void tambahDiAwal(List, int value)
42 {
43     // Buat node baru
44     Node* newNode = new Node(value);
45     // Jika list kosong
46     if (head == NULL)
47     {
48         head = newNode;
49     }
50     // Jika list tidak kosong
51     else
52     {
53         newNode->next = head;
54         head = newNode;
55     }
56 }
57
58 // Fungsi untuk menambahkan elemen di akhir list
59 void tambahDiAkhir(List, int value)
60 {
61     // Buat node baru
62     Node* newNode = new Node(value);
63     // Jika list kosong
64     if (head == NULL)
65     {
66         head = newNode;
67     }
68     // Jika list tidak kosong
69     else
70     {
71         // Temukan elemen terakhir
72         Node* temp = head;
73         while (temp->next != NULL)
74         {
75             temp = temp->next;
76         }
77         temp->next = newNode;
78     }
79 }
80
81 // Fungsi untuk menampilkan list
82 void tampilkanList(List)
83 {
84     if (head == NULL)
85     {
86         cout << "List kosong" << endl;
87         return;
88     }
89     // Iterasi untuk menampilkan list
90     Node* temp = head;
91     while (temp != NULL)
92     {
93         cout << temp->data << " ";
94         temp = temp->next;
95     }
96     cout << endl;
97 }
98
99 // Fungsi untuk menghitung jumlah elemen
100 int hitungJumlahElemen(List)
101 {
102     int count = 0;
103     Node* temp = head;
104     while (temp != NULL)
105     {
106         count++;
107         temp = temp->next;
108     }
109     return count;
110 }
111
112 // Fungsi untuk menghapus semua elemen dari list dan menginisialisasi kembali
113 void resetList(List)
114 {
115     // Hapus semua elemen
116     hapusSemuaElemen(List);
117     // Inisialisasi list baru
118     head = NULL;
119 }
120
121 // Fungsi untuk menampilkan menu
122 void tampilkanMenu()
123 {
124     cout << "Menu" << endl;
125     cout << "1. Tambah di awal" << endl;
126     cout << "2. Tambah di akhir" << endl;
127     cout << "3. Cari elemen" << endl;
128     cout << "4. Hapus semua" << endl;
129     cout << "5. Hitung jumlah" << endl;
130     cout << "6. Reset list" << endl;
131     cout << "7. Exit" << endl;
132 }

```

Program C++ di atas adalah implementasi **single linked list** sederhana yang mencakup beberapa operasi dasar, seperti menambahkan elemen di awal dan akhir list, menampilkan elemen, menghitung jumlah elemen, serta menghapus seluruh elemen list.

**Struct Node:** Digunakan untuk mendefinisikan node pada linked list, yang terdiri dari dua komponen:

- data: Menyimpan nilai elemen.

- next: Pointer yang menunjuk ke node berikutnya.

**alokasi():** Fungsi untuk mengalokasikan memori bagi node baru. Jika berhasil, node baru akan diisi dengan nilai yang diberikan dan pointer next diatur ke nullptr.

**dealokasi():** Fungsi untuk menghapus memori yang digunakan oleh node. Ini digunakan saat node dihapus.

**isListEmpty():** Fungsi untuk memeriksa apakah linked list kosong dengan memeriksa apakah head bernilai nullptr.

## 4. UNGUIDED

### 1.

```

1  #include <iostream>
2  using namespace std;
3
4
5  struct Node {
6      int data;
7      Node* next;
8  };
9
10
11 void insertDepan(Node*& head, int nilai) {
12     Node* newNode = new Node();
13     newNode->data = nilai;
14     newNode->next = head;
15     head = newNode;
16 }
17
18
19 void insertBelakang(Node*& head, int nilai) {
20     Node* newNode = new Node();
21     newNode->data = nilai;
22     newNode->next = nullptr;
23
24     if (head == nullptr) {
25         head = newNode;
26     } else {
27         Node* temp = head;
28         while (temp->next != nullptr) {
29             temp = temp->next;
30         }
31         temp->next = newNode;
32     }
33 }
34
35
36 void cetakLinkedList(Node* head) {
37     Node* temp = head;
38     while (temp != nullptr) {
39         cout << temp->data;
40         if (temp->next != nullptr) {
41             cout << " -> ";
42         }
43         temp = temp->next;
44     }
45     cout << endl;
46 }
47
48 int main() {
49     Node* head = nullptr;
50
51     insertDepan(head, 10);
52     insertBelakang(head, 20);
53     insertDepan(head, 5);
54
55     cout << "Isi linked list: ";
56     cetakLinkedList(head);
57     return 0;
58 }

```

Output:

```

[Running] cd "D:\pemrograman java\" && g++ unguidedilaprak4.cpp -o unguidedilaprak4 && "D:\pemrograman java\unguidedilaprak4
Isi linked list: 5 -> 10 -> 20

```

2.

```

1  // 1. Menghitung jumlah semua
2  int sum = 0;
3
4  // 2. Mendefinisikan variabel untuk menyimpan hasil
5  int hasil = 0;
6
7  // 3. Menghitung jumlah semua bilangan dari 1 hingga 10
8  for (int i = 1; i <= 10; i++) {
9      sum += i;
10 }
11
12 // 4. Menghitung jumlah semua bilangan dari 1 hingga 10
13 int sum2 = 0;
14 for (int i = 1; i <= 10; i++) {
15     sum2 += i;
16 }
17
18 // 5. Menghitung jumlah semua bilangan dari 1 hingga 10
19 int sum3 = 0;
20 for (int i = 1; i <= 10; i++) {
21     sum3 += i;
22 }
23
24 // 6. Menghitung jumlah semua bilangan dari 1 hingga 10
25 int sum4 = 0;
26 for (int i = 1; i <= 10; i++) {
27     sum4 += i;
28 }
29
30 // 7. Menghitung jumlah semua bilangan dari 1 hingga 10
31 int sum5 = 0;
32 for (int i = 1; i <= 10; i++) {
33     sum5 += i;
34 }
35
36 // 8. Menghitung jumlah semua bilangan dari 1 hingga 10
37 int sum6 = 0;
38 for (int i = 1; i <= 10; i++) {
39     sum6 += i;
40 }
41
42 // 9. Menghitung jumlah semua bilangan dari 1 hingga 10
43 int sum7 = 0;
44 for (int i = 1; i <= 10; i++) {
45     sum7 += i;
46 }
47
48 // 10. Menghitung jumlah semua bilangan dari 1 hingga 10
49 int sum8 = 0;
50 for (int i = 1; i <= 10; i++) {
51     sum8 += i;
52 }
53
54 // 11. Menghitung jumlah semua bilangan dari 1 hingga 10
55 int sum9 = 0;
56 for (int i = 1; i <= 10; i++) {
57     sum9 += i;
58 }
59
60 // 12. Menghitung jumlah semua bilangan dari 1 hingga 10
61 int sum10 = 0;
62 for (int i = 1; i <= 10; i++) {
63     sum10 += i;
64 }
65
66 // 13. Menghitung jumlah semua bilangan dari 1 hingga 10
67 int sum11 = 0;
68 for (int i = 1; i <= 10; i++) {
69     sum11 += i;
70 }
71
72 // 14. Menghitung jumlah semua bilangan dari 1 hingga 10
73 int sum12 = 0;
74 for (int i = 1; i <= 10; i++) {
75     sum12 += i;
76 }
77
78 // 15. Menghitung jumlah semua bilangan dari 1 hingga 10
79 int sum13 = 0;
80 for (int i = 1; i <= 10; i++) {
81     sum13 += i;
82 }
83
84 // 16. Menghitung jumlah semua bilangan dari 1 hingga 10
85 int sum14 = 0;
86 for (int i = 1; i <= 10; i++) {
87     sum14 += i;
88 }
89
90 // 17. Menghitung jumlah semua bilangan dari 1 hingga 10
91 int sum15 = 0;
92 for (int i = 1; i <= 10; i++) {
93     sum15 += i;
94 }
95
96 // 18. Menghitung jumlah semua bilangan dari 1 hingga 10
97 int sum16 = 0;
98 for (int i = 1; i <= 10; i++) {
99     sum16 += i;
100 }
101
102 // 19. Menghitung jumlah semua bilangan dari 1 hingga 10
103 int sum17 = 0;
104 for (int i = 1; i <= 10; i++) {
105     sum17 += i;
106 }
107
108 // 20. Menghitung jumlah semua bilangan dari 1 hingga 10
109 int sum18 = 0;
110 for (int i = 1; i <= 10; i++) {
111     sum18 += i;
112 }
113
114 // 21. Menghitung jumlah semua bilangan dari 1 hingga 10
115 int sum19 = 0;
116 for (int i = 1; i <= 10; i++) {
117     sum19 += i;
118 }
119
120 // 22. Menghitung jumlah semua bilangan dari 1 hingga 10
121 int sum20 = 0;
122 for (int i = 1; i <= 10; i++) {
123     sum20 += i;
124 }
125
126 // 23. Menghitung jumlah semua bilangan dari 1 hingga 10
127 int sum21 = 0;
128 for (int i = 1; i <= 10; i++) {
129     sum21 += i;
130 }
131
132 // 24. Menghitung jumlah semua bilangan dari 1 hingga 10
133 int sum22 = 0;
134 for (int i = 1; i <= 10; i++) {
135     sum22 += i;
136 }
137
138 // 25. Menghitung jumlah semua bilangan dari 1 hingga 10
139 int sum23 = 0;
140 for (int i = 1; i <= 10; i++) {
141     sum23 += i;
142 }
143
144 // 26. Menghitung jumlah semua bilangan dari 1 hingga 10
145 int sum24 = 0;
146 for (int i = 1; i <= 10; i++) {
147     sum24 += i;
148 }
149
150 // 27. Menghitung jumlah semua bilangan dari 1 hingga 10
151 int sum25 = 0;
152 for (int i = 1; i <= 10; i++) {
153     sum25 += i;
154 }
155
156 // 28. Menghitung jumlah semua bilangan dari 1 hingga 10
157 int sum26 = 0;
158 for (int i = 1; i <= 10; i++) {
159     sum26 += i;
160 }
161
162 // 29. Menghitung jumlah semua bilangan dari 1 hingga 10
163 int sum27 = 0;
164 for (int i = 1; i <= 10; i++) {
165     sum27 += i;
166 }
167
168 // 30. Menghitung jumlah semua bilangan dari 1 hingga 10
169 int sum28 = 0;
170 for (int i = 1; i <= 10; i++) {
171     sum28 += i;
172 }
173
174 // 31. Menghitung jumlah semua bilangan dari 1 hingga 10
175 int sum29 = 0;
176 for (int i = 1; i <= 10; i++) {
177     sum29 += i;
178 }
179
180 // 32. Menghitung jumlah semua bilangan dari 1 hingga 10
181 int sum30 = 0;
182 for (int i = 1; i <= 10; i++) {
183     sum30 += i;
184 }
185
186 // 33. Menghitung jumlah semua bilangan dari 1 hingga 10
187 int sum31 = 0;
188 for (int i = 1; i <= 10; i++) {
189     sum31 += i;
190 }
191
192 // 34. Menghitung jumlah semua bilangan dari 1 hingga 10
193 int sum32 = 0;
194 for (int i = 1; i <= 10; i++) {
195     sum32 += i;
196 }
197
198 // 35. Menghitung jumlah semua bilangan dari 1 hingga 10
199 int sum33 = 0;
200 for (int i = 1; i <= 10; i++) {
201     sum33 += i;
202 }
203
204 // 36. Menghitung jumlah semua bilangan dari 1 hingga 10
205 int sum34 = 0;
206 for (int i = 1; i <= 10; i++) {
207     sum34 += i;
208 }
209
210 // 37. Menghitung jumlah semua bilangan dari 1 hingga 10
211 int sum35 = 0;
212 for (int i = 1; i <= 10; i++) {
213     sum35 += i;
214 }
215
216 // 38. Menghitung jumlah semua bilangan dari 1 hingga 10
217 int sum36 = 0;
218 for (int i = 1; i <= 10; i++) {
219     sum36 += i;
220 }
221
222 // 39. Menghitung jumlah semua bilangan dari 1 hingga 10
223 int sum37 = 0;
224 for (int i = 1; i <= 10; i++) {
225     sum37 += i;
226 }
227
228 // 40. Menghitung jumlah semua bilangan dari 1 hingga 10
229 int sum38 = 0;
230 for (int i = 1; i <= 10; i++) {
231     sum38 += i;
232 }
233
234 // 41. Menghitung jumlah semua bilangan dari 1 hingga 10
235 int sum39 = 0;
236 for (int i = 1; i <= 10; i++) {
237     sum39 += i;
238 }
239
240 // 42. Menghitung jumlah semua bilangan dari 1 hingga 10
241 int sum40 = 0;
242 for (int i = 1; i <= 10; i++) {
243     sum40 += i;
244 }
245
246 // 43. Menghitung jumlah semua bilangan dari 1 hingga 10
247 int sum41 = 0;
248 for (int i = 1; i <= 10; i++) {
249     sum41 += i;
250 }
251
252 // 44. Menghitung jumlah semua bilangan dari 1 hingga 10
253 int sum42 = 0;
254 for (int i = 1; i <= 10; i++) {
255     sum42 += i;
256 }
257
258 // 45. Menghitung jumlah semua bilangan dari 1 hingga 10
259 int sum43 = 0;
260 for (int i = 1; i <= 10; i++) {
261     sum43 += i;
262 }
263
264 // 46. Menghitung jumlah semua bilangan dari 1 hingga 10
265 int sum44 = 0;
266 for (int i = 1; i <= 10; i++) {
267     sum44 += i;
268 }
269
270 // 47. Menghitung jumlah semua bilangan dari 1 hingga 10
271 int sum45 = 0;
272 for (int i = 1; i <= 10; i++) {
273     sum45 += i;
274 }
275
276 // 48. Menghitung jumlah semua bilangan dari 1 hingga 10
277 int sum46 = 0;
278 for (int i = 1; i <= 10; i++) {
279     sum46 += i;
280 }
281
282 // 49. Menghitung jumlah semua bilangan dari 1 hingga 10
283 int sum47 = 0;
284 for (int i = 1; i <= 10; i++) {
285     sum47 += i;
286 }
287
288 // 50. Menghitung jumlah semua bilangan dari 1 hingga 10
289 int sum48 = 0;
290 for (int i = 1; i <= 10; i++) {
291     sum48 += i;
292 }
293
294 // 51. Menghitung jumlah semua bilangan dari 1 hingga 10
295 int sum49 = 0;
296 for (int i = 1; i <= 10; i++) {
297     sum49 += i;
298 }
299
300 // 52. Menghitung jumlah semua bilangan dari 1 hingga 10
301 int sum50 = 0;
302 for (int i = 1; i <= 10; i++) {
303     sum50 += i;
304 }
305
306 // 53. Menghitung jumlah semua bilangan dari 1 hingga 10
307 int sum51 = 0;
308 for (int i = 1; i <= 10; i++) {
309     sum51 += i;
310 }
311
312 // 54. Menghitung jumlah semua bilangan dari 1 hingga 10
313 int sum52 = 0;
314 for (int i = 1; i <= 10; i++) {
315     sum52 += i;
316 }
317
318 // 55. Menghitung jumlah semua bilangan dari 1 hingga 10
319 int sum53 = 0;
320 for (int i = 1; i <= 10; i++) {
321     sum53 += i;
322 }
323
324 // 56. Menghitung jumlah semua bilangan dari 1 hingga 10
325 int sum54 = 0;
326 for (int i =
```

Output:

```
[Running] cd "d:\pemrograman java\" && g++ unguidedno2laprak4.cpp -o unguidedno2laprak4 && "d:\pemrograman java\unguidedno2laprak4
5 -> 20
```

3.

```
1 #include <iostream>
2 using namespace std;
3
4 struct Node {
5     int data;
6     Node* next;
7 }
8
9 void InsertDepan(Node* head, int nilai) {
10     Node* newnode = new Node();
11     newnode->data = nilai;
12     newnode->next = head;
13     head = newnode;
14 }
15
16 void InsertBelakang(Node* head, int nilai) {
17     Node* newnode = new Node();
18     newnode->data = nilai;
19     newnode->next = nullptr;
20
21     if (head == nullptr) {
22         head = newnode;
23     } else {
24         Node* temp = head;
25         while (temp->next != nullptr) {
26             temp = temp->next;
27         }
28         temp->next = newnode;
29     }
30 }
31
32 Node* cariNode(Node* head, int nilai) {
33     Node* temp = head;
34     while (temp != nullptr) {
35         if (temp->data == nilai) {
36             return temp;
37         }
38         temp = temp->next;
39     }
40     return nullptr;
41 }
42
43 int hitungPanjang(Node* head) {
44     int panjang = 0;
45     Node* temp = head;
46     while (temp != nullptr) {
47         panjang++;
48         temp = temp->next;
49     }
50     return panjang;
51 }
52
53 void cetakLinkedlist(Node* head) {
54     Node* temp = head;
55     while (temp != nullptr) {
56         cout << temp->data;
57         if (temp->next != nullptr) {
58             cout << " -> ";
59         }
60         temp = temp->next;
61     }
62     cout << endl;
63 }
64
65 int main() {
66     Node* head = nullptr;
67
68     InsertDepan(head, 10);
69     InsertBelakang(head, 20);
70     InsertDepan(head, 5);
71
72     cout << "Ini linked list: ";
73     cetakLinkedlist(head);
74
75     int nilaiCari = 20;
76     if (cariNode(head, nilaiCari)) {
77         cout << "Node dengan nilai " << nilaiCari << " ditemukan." << endl;
78     } else {
79         cout << "Node dengan nilai " << nilaiCari << " tidak ditemukan." << endl;
80     }
81
82     int panjang = hitungPanjang(head);
83     cout << "Panjang linked list: " << panjang << endl;
84
85     return 0;
86 }
```

Output:

```
Node dengan nilai 20 ditemukan.
Panjang linked list: 3
```



## **5.KESIMPULAN**

Single linked list memberikan fleksibilitas yang lebih tinggi dibandingkan array dalam hal pengelolaan memori dinamis. Meskipun memiliki beberapa kekurangan, seperti akses yang lebih lambat dan penggunaan memori tambahan untuk pointer, linked list sangat efisien dalam operasi penambahan dan penghapusan elemen. Memahami dan mengimplementasikan linked list dengan benar adalah langkah penting dalam pemahaman dasar struktur data dan algoritma.