



LEMBAR ASISTENSI  
PRAKTIKUM STRUKTUR DATA  
LABORATORIUM TEKNIK KOMPUTER  
JURUSAN TEKNIK ELEKTRO FAKULTAS TEKNIK  
UNIVERSITAS LAMPUNG

---

Judul Praktikum : LINKED LIST  
Praktikan : Alvin Reihansyah Makarim (2115061083)  
Asisten : Asha Imalia Zahra (2015061006)  
Bagus Wahyu Pratomo (2015061003)

No.	Catatan	Tanggal	Paraf

Bandar Lampung, 2022

.....  
NPM.

## I. JUDUL PERCOBAAN

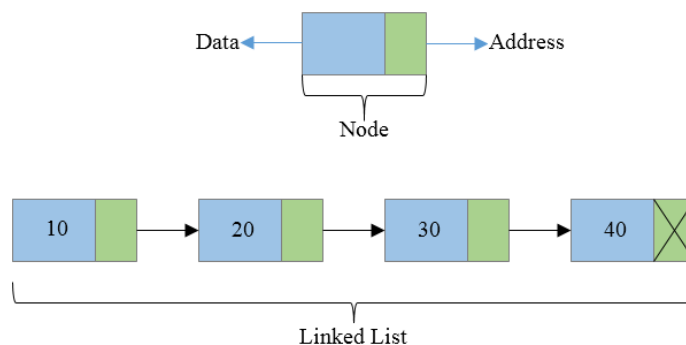
### LINKED LIST

## II. TUJUAN PERCOBAAN

Adapun tujuan dari percobaan ini adalah sebagai berikut :

1. Mahasiswa dapat memahami linked list
2. Mahasiswa dapat membuat linked list

## III. TEORI DASAR



Gambar 3.1 Linked list

Linked list adalah salah satu bentuk struktur data linear yang berbentuk menyerupai rantai bersimpul dimana setiap simpulnya menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Simpul pertama disebut dengan head, sedangkan simpul terakhir disebut tail. Jika linked list kosong, maka nilai pointer pada head dan pointer berikutnya hingga tail akan menunjuk ke NULL.

Terdapat 3 jenis linked list, yaitu :

- Single Linked List

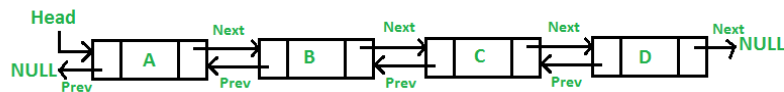


Gambar 3.2 Single linked list

Single linked list adalah bentuk paling sederhana dari linked list. Single linked list adalah bentuk linked list yang mempunyai satu pointer ke simpul lainnya. Pembentukan linked list tunggal memerlukan :

1. Deklarasi tipe simpul
2. Deklarasi variabel pointer penunjuk awal Linked list
3. Pembentukan simpul baru
4. Pengaitan simpul baru ke Linked list yang telah terbentuk

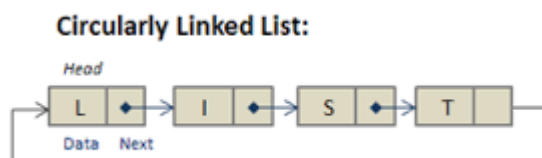
- Double Linked List



Gambar 3.3 Double Linked List

Double linked list adalah jenis linked list yang mempunyai dua pointer, yaitu pointer yang menunjuk ke data sebelumnya dan pointer yang menunjuk ke data berikutnya.

- Circular Linked List



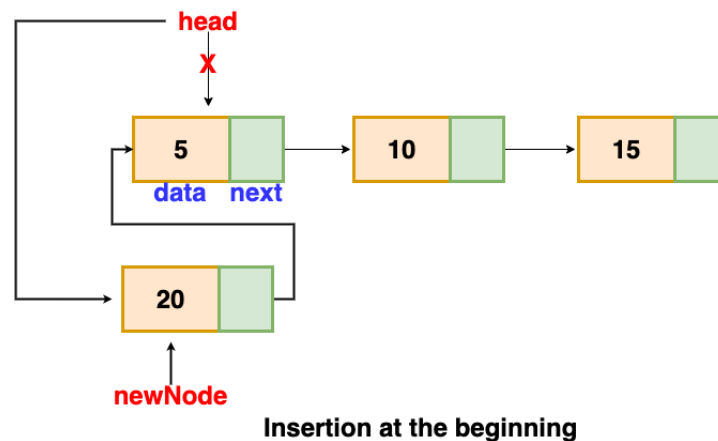
Gambar 3.4 Circular linked list

Circular linked list adalah linked list unidirectional. Kita hanya dapat melintasinya dalam satu arah. Tetapi jenis linked list ini memiliki simpul terakhir yang menunjuk ke simpul kepala.

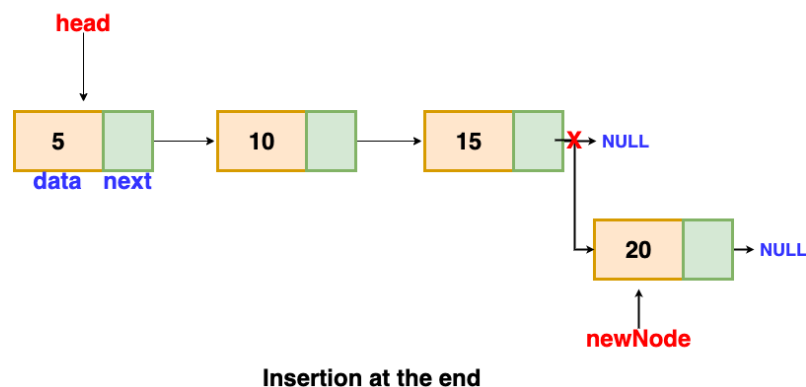
Linked list tidak sama dengan array. Array merupakan struktur data berbasis indeks, setiap elemen dalam array terkait dengan indeks tersebut, sedangkan linked list bergantung pada referensi. Node-node dalam linked list terdiri dari data dan referensi ke elemen data yang lain. Perbedaan yang kedua, array adalah kumpulan objek data yang mirip satu sama lain dan disimpan di lokasi memori secara berurutan, sedangkan linked list merupakan sekumpulan data yang berisi urutan elemen dalam strukturnya. Setiap elemen saling terkait dengan elemen berikutnya.

Terdapat 5 operasi yang terdapat pada linked list, yaitu :

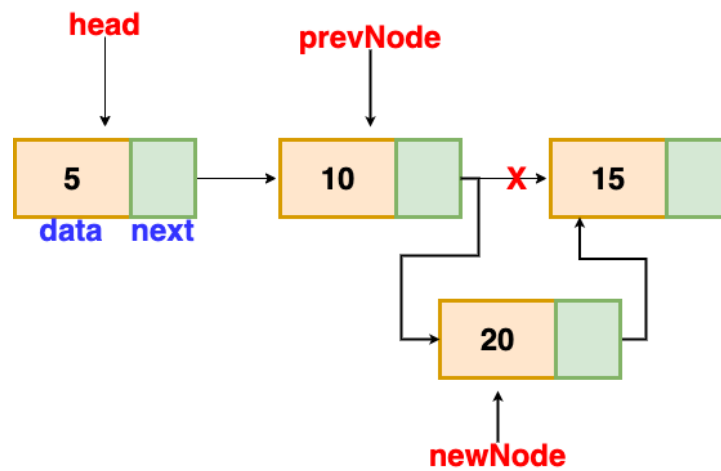
- a) Menambah simpul : Kita dapat menambahkan simpul dengan meletakkannya pada bagian paling akhir atau paling depan dari linked list.



Gambar 3.5 Menambahkan data pada bagian depan Linked List



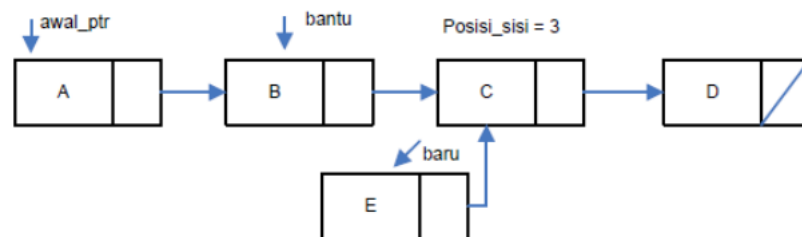
Gambar 3.6 Menambahkan data pada bagian belakang Linked List



**Insertion after a given node**

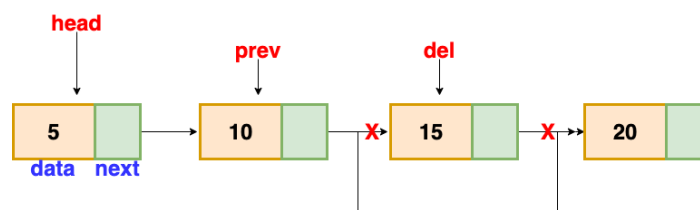
Gambar 3.7 Menambahkan data diantara simpul

b) Menyisipkan data : Kita dapat menyisipkan data pada simpul yang sudah ada.



Gambar 3.8 Menyisipkan data

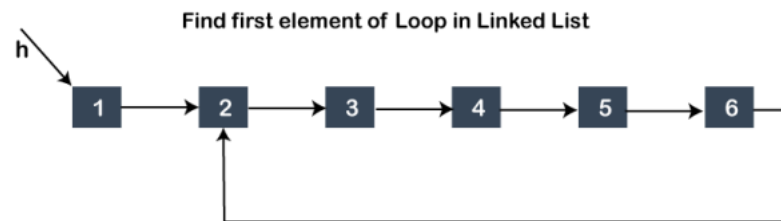
c) Menghapus data : Kita dapat menghapus data yang ada pada simpul bagian depan, belakang, atau ditengah dengan syarat harus sesudah simpul yang ditunjuk oleh suatu pointer.



**Deleting a Node in Linked List**

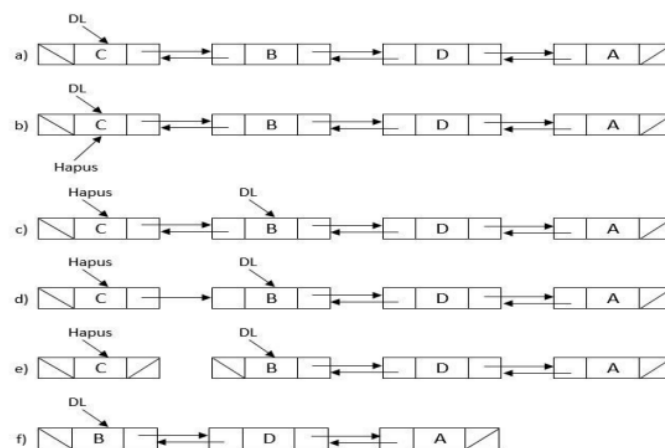
Gambar 3.9 Menghapus data

- d) Mencari informasi : Kita dapat mencari informasi di dalam simpul dengan cara menerapkan testing untuk menentukan ada tidaknya data yang dicari.



Gambar 3.10 Mencari informasi

- e) Mencetak simpul : Kita dapat mencetak simpul dengan cara membacanya secara maju atau mundur.



Gambar3.11 Mencetak simpul

Kelebihan dari linked list dibandingkan struktur data lainnya diantaranya struktur datanya yang dinamis, tidak boros memori, pengimplamentasian yang mudah, dan operasi-operasi yang dimiliki linked list itu sendiri. Struktur data linked list bersifat dinamis sehingga dapat bertambah dan menyusut saat runtime dengan mengalokasikan dan membatalkan akses memori. Kemudian pemanfaatan memori dari linked list juga lebih baik karena ukurannya yang berkurang atau bertambah pada runtime sehingga tidak ada pemborosan memori untuk alokasi data kosong. Struktur data linier seperti stack dan queue juga seringkali lebih mudah diimplementasikan menggunakan linked list.

#### IV. PROSEDUR PERCOBAAN

Adapun source code untuk percobaan ini adalah sebagai berikut :

##### 4.1. Percobaan 4-1: Linked List Beginning Insertion

```
1  #include <iostream>
2  using namespace std;
3
4  struct node
5  {
6      int info;
7      node *next;
8  } * start, *newptr, *save, *ptr;
9  node *create_new_node(int);
10 void insert_at_beg(node *);
11 void display(node *);
12 int main()
13 {
14     start = NULL;
15     int inf;
16     char ch = 'y';
17     while (ch == 'y' || ch == 'Y')
18     {
19         cout << "Masukkan nilai untuk node baru: ";
20         cin >> inf;
21         cout << "Membuat node baru..." << endl;
22         newptr = create_new_node(inf);
23         if (newptr != NULL)
24         {
25             cout << "Berhasil membuat node baru..." << endl;
26         }
27         else
28         {
29             cout << "Maaf, tidak dapat membuat node baru";
30             return 0;
31         }
32         cout << "Memasukkan node pada bagian awal list..." << endl;
33         insert_at_beg(newptr);
34         cout << "Node berhasil dimasukkan di bagian awal list..." << endl;
35         cout << "List: ";
36         display(start);
37         cout << "Mau membuat node baru? (y/n) ";
38         cin >> ch;
39     }
40     return 0;
41 }
42 node *create_new_node(int n)
43 {
44     ptr = new node;
45     ptr->info = n;
46     ptr->next = NULL;
47     return ptr;
48 }
49 void insert_at_beg(node *np)
50 {
51     if (start == NULL)
52     {
53         start = np;
54     }
55     else
56     {
57         save = start;
58         start = np;
59         np->next = save;
60     }
61 }
62 void display(node *np)
63 {
64     while (np != NULL)
65     {
66         cout << np->info << " -> ";
67         np = np->next;
68     }
69     cout << "!!!\n";
70 }
```

Gambar 4.1 Linked List Insertion

## 4.2 Percobaan 4-2: Linked List End Insertion

```
1  #include <iostream>
2  using namespace std;
3
4  struct node
5  {
6      int info;
7      node *next;
8  } * start, *newptr, *save, *ptr, *rear;
9  node *create_new_node(int);
10 void insert_in_end(node *);
11 void display(node *);
12 int main()
13 {
14     start = rear = NULL;
15     int inf;
16     char ch = 'y';
17     while (ch == 'y' || ch == 'Y')
18     {
19         cout << "Masukkan nilai untuk node baru: ";
20         cin >> inf;
21         cout << "Membuat node baru..." << endl;
22         newptr = create_new_node(inf);
23         if (newptr != NULL)
24         {
25             cout << "Berhasil membuat node baru..." << endl;
26         }
27         else
28         {
29             cout << "Maaf, tidak dapat membuat node baru";
30             return 0;
31         }
32         cout << "Memasukkan node pada bagian akhir list..." << endl;
33         insert_in_end(newptr);
34         cout << "Node berhasil dimasukkan di bagian akhir list..." << endl;
35         cout << "List: ";
36         display(start);
37         cout << "Mau membuat node baru? (y/n) ";
38         cin >> ch;
39     }
40     return 0;
41 }
42 node *create_new_node(int n)
43 {
44     ptr = new node;
45     ptr->info = n;
46     ptr->next = NULL;
47     return ptr;
48 }
49 void insert_in_end(node *np)
50 {
51     if (start == NULL)
52     {
53         start = rear = np;
54     }
55     else
56     {
57         rear->next = np;
58         rear = np;
59     }
60 }
61 void display(node *np)
62 {
63     while (np != NULL)
64     {
65         cout << np->info << " -> ";
66         np = np->next;
67     }
68     cout << "!!!\n";
69 }
```

Gambar 4.2 Linked List End Insertion



### 4.3 Percobaan 4-3: Linked List Delete Node

```
1 #include <iostream>
2 using namespace std;
3 struct node
4 {
5     int info;
6     node *next;
7 } * start, *newptr, *save, *ptr, *rear;
8 node *create_new_node(int);
9 void insert_node(node *);
10 void display(node *);
11 void delete_node();
12 int main()
13 {
14     start = rear = NULL;
15     int inf;
16     char ch = 'y';
17     while (ch == 'y' || ch == 'Y')
18     {
19         cout << "Masukkan nilai untuk node baru: ";
20         cin >> inf;
21         cout << "Membuat node baru..." << endl;
22         newptr = create_new_node(inf);
23         if (newptr != NULL)
24         {
25             cout << "Berhasil membuat node baru..." << endl;
26         }
27         else
28         {
29             cout << "Maaf, tidak dapat membuat node baru";
30             return 0;
31         }
32         cout << "Memasukkan node pada bagian akhir list..." << endl;
33         insert_node(newptr);
34         cout << "Node berhasil dimasukkan di bagian akhir list..." << endl;
35         cout << "List: ";
36         display(start);
37         cout << "Mau membuat node baru? (y/n) ";
38         cin >> ch;
39     }
40     do
41     {
42         cout << "List: ";
43         display(start);
44         cout << "Mau menghapus node pertama? (y/n) ";
45         cin >> ch;
46         if (ch == 'y' || ch == 'Y')
47         {
48             delete_node();
49         }
50     } while (ch == 'y' || ch == 'Y');
51     return 0;
52 }
53 node *create_new_node(int n)
54 {
55     ptr = new node;
56     ptr->info = n;
57     ptr->next = NULL;
58     return ptr;
59 }
60 void insert_node(node *np)
61 {
62     if (start == NULL)
63     {
64         start = rear = np;
65     }
66     else
67     {
68         rear->next = np;
69         rear = np;
70     }
71 }
72 void delete_node()
73 {
74     if (start == NULL)
75     {
76         cout << "Underflow...!!" << endl;
77     }
78     else
79     {
80         ptr = start;
81         start = start->next;
82     }
83 }
84 void display(node *np)
85 {
86     while (np != NULL)
87     {
88         cout << np->info << " -> ";
89         np = np->next;
90     }
91     cout << "!!\n";
92 }
```

Gambar 4.3 Linked List Delete Node

#### 4.4 Percobaan 4-4: Linked List Traversal

```
1 #include <iostream>
2 using namespace std;
3 struct node
4 {
5     int info;
6     node *next;
7 } *start, *newptr, *save, *ptr, *rear;
8 node *create_new_node(int);
9 void insert_node(node *);
10 void travers(node *);
11 int main()
12 {
13     start = rear = NULL;
14     int inf;
15     char ch = 'y';
16     while (ch == 'y' || ch == 'Y')
17     {
18         cout << "Masukkan nilai untuk node baru: ";
19         cin >> inf;
20         cout << "Membuat node baru..." << endl;
21         newptr = create_new_node(inf);
22         if (newptr != NULL)
23         {
24             cout << "Berhasil membuat node baru..." << endl;
25         }
26         else
27         {
28             cout << "Maaf, tidak dapat membuat node baru";
29             return 0;
30         }
31         cout << "Memasukkan node pada bagian akhir list..." << endl;
32         insert_node(newptr);
33         cout << "Node berhasil dimasukkan di bagian akhir list..." << endl;
34         cout << "Mau membuat node baru? (y/n) ";
35         cin >> ch;
36     }
37     cout << "List: ";
38     travers(start);
39     if (start != NULL)
40     {
41         cout << "Mengakses nilai pada satu node: ";
42         cout << start->info << endl;
43         cout << "Alamat dari node ini: ";
44         cout << &start << endl;
45         cout << "Alamat dari node selanjutnya: ";
46         cout << start->next << endl;
47     }
48     if (start->next != NULL)
49     {
50         cout << "Mengakses nilai pada node selanjutnya: ";
51         cout << start->next->info << endl;
52         cout << "Alamat dari node ini: ";
53         cout << start->next << endl;
54         cout << "Alamat dari node selanjutnya: ";
55         cout << start->next->next << endl;
56     }
57     return 0;
58 }
59 node *create_new_node(int n)
60 {
61     ptr = new node;
62     ptr->info = n;
63     ptr->next = NULL;
64     return ptr;
65 }
66 void insert_node(node *np)
67 {
68     if (start == NULL)
69     {
70         start = rear = np;
71     }
72     else
73     {
74         rear->next = np;
75         rear = np;
76     }
77 }
78 void travers(node *np)
79 {
80     while (np != NULL)
81     {
82         cout << np->info << " -> ";
83         np = np->next;
84     }
85     cout << " !!\n";
86 }
```

Gambar 4.4 Linked List Traversal

## V. PEMBAHASAN

Adapun source code percobaan ini adalah sebagai berikut :

### 5.1. Percobaan 4-1: Linked List Beginning Insertion

#### 5.1.a Source code

```
1 #include <iostream>
2 using namespace std;
3
4 struct node
5 {
6     int info;
7     node *next;
8 } * start, *newptr, *save, *ptr;
9 node *create_new_node(int);
10 void insert_at_beg(node *);
11 void display(node *);
12 int main()
13 {
14     start = NULL;
15     int inf;
16     char ch = 'y';
17     while (ch == 'y' || ch == 'Y')
18     {
19         cout << "Masukkan nilai untuk node baru: ";
20         cin >> inf;
21         cout << "Membuat node baru..." << endl;
22         newptr = create_new_node(inf);
23         if (newptr != NULL)
24         {
25             cout << "Berhasil membuat node baru..." << endl;
26         }
27         else
28         {
29             cout << "Maaf, tidak dapat membuat node baru";
30             return 0;
31         }
32         cout << "Memasukkan node pada bagian awal list..." << endl;
33         insert_at_beg(newptr);
34         cout << "Node berhasil dimasukkan di bagian awal list..." << endl;
35         cout << "List: ";
36         display(start);
37         cout << "Mau membuat node baru? (y/n) ";
38         cin >> ch;
39     }
40     return 0;
41 }
42 node *create_new_node(int n)
43 {
44     ptr = new node;
45     ptr->info = n;
46     ptr->next = NULL;
47     return ptr;
48 }
49 void insert_at_beg(node *np)
50 {
51     if (start == NULL)
52     {
53         start = np;
54     }
55     else
56     {
57         save = start;
58         start = np;
59         np->next = save;
60     }
61 }
62 void display(node *np)
63 {
64     while (np != NULL)
65     {
66         cout << np->info << " -> ";
67         np = np->next;
68     }
69     cout << "!!!\n";
70 }
```

Gambar 5.1.a Source Code Linked List Insertion

Berdasarkan gambar 5.1.a yang merupakan source code dari program linked list insertion, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 4 terdapat deklarasi structure dengan nama node. Di dalamnya, terdapat dua field, yaitu integer info pada baris ke 6 dan node \*next pada baris ke 7. Selain itu terdapat penggunaan start, newptr, save, dan ptr di dalam structure node. Kemudian terdapat inisialisasi create\_new\_node dengan tipe data integer pada baris ke 9. Kemudian pada baris ke 10 terdapat inisialisasi void function insert\_at\_beg dengan parameter node. Pada baris ke 11 terdapat deklarasi void function display dengan parameter node. Kemudian pada baris ke 12 terdapat deklarasi fungsi utama. Pada baris ke 14 terdapat inisialisasi nilai start = NULL. Pada baris ke 15 terdapat deklarasi variabel integer dengan nama inf. Pada baris ke 16 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 17 terdapat perulangan while dengan kondisi ch == y atau ch == Y. Di dalam perulangan tersebut, pada baris ke 19 terdapat perintah untuk menampilkan pesan “Masukkan nilai untuk node baru: “. Pada baris ke 20 terdapat perintah agar user memasukkan nilai ke variabel inf. Pada baris ke 21 terdapat perintah untuk menampilkan pesan “Membuat node baru”. Pada baris ke 22 terdapat inisialisasi nilai newptr = nilai dari fungsi create\_new\_node dengan argumen inf. Pada baris ke 23 terdapat percabangan if dengan kondisi jika newptr != NULL, maka tampilkan pesan “Berhasil membuat node baru”. Jika tidak terpenuhi, maka tampilkan pesan “Maaf, tidak dapat membuat node baru” dan program berakhir dengan return 0. Pada baris ke 32 terdapat perintah untuk menampilkan pesan “Memasukkan node pada bagian awal list...” Pada baris ke 33 terdapat perintah untuk menjalankan fungsi insert\_at\_beg dengan argument nilai dari newptr. Pada baris ke 34, terdapat perintah untuk menampilkan pesan “Node berhasil dimasukkan di bagian awal list...”. Pada baris ke 35 terdapat perintah untuk menampilkan pesan “List: “. Pada baris ke 36 terdapat perintah untuk menjalankan fungsi display dengan argument start. Pada baris ke 37, terdapat perintah untuk menampilkan pesan “Mau membuat node baru?” Kemudian pada baris ke 38 terdapat perintah agar user memasukkan nilai ke dalam variabel ch. Pada baris ke 40 terdapat return 0 yang menyatakan hasil

keluaran dari fungsi main() bahwa program berakhir dengan normal. Pada baris ke 42 terdapat inisialisasi dari node create\_new\_node dengan parameter int n. Pada baris ke 44 terdapat inisialisasi nilai ptr dengan nilai dari new node. Pada baris ke 45 terdapat assignment nilai ptr info dengan n. Pada baris ke 46 terdapat assignment nilai ptr next dengan NULL. Fungsi kemudian mengembalikan nilai ptr. Pada baris ke 49 terdapat inisialisasi fungsi insert\_at\_beg dengan parameter node np. Pada baris ke 51, terdapat percabangan if dengan kondisi jika start = Null maka inisialisasi nilai start dengan np, jika tidak terpenuhi, maka inisialisasi nilai save dengan start, inisialisasi nilai start dengan np, dan assignment nilai np next dengan nilai dari save. Pada baris ke 62 terdapat inisialisasi fungsi display dengan parameter node np. Pada baris ke 64 terdapat perulangan while dengan kondisi np != NULL), maka program akan menampilkan nilai dari np info dan kemudian menginisialisasi nilai dari np dengan np next. Kemudian pada baris ke 69 terdapat perintah untuk menampilkan pesan !! dan berpindah baris.

### 5.1.b Output

```
($?) { g++ kode4-3-1.cpp -o kode4-3-1 } ; if ($?) { . \kode4-3-1 }
Masukkan nilai untuk node baru: 12
Membuat node baru...
Berhasil membuat node baru...
Memasukkan node pada bagian awal list...
Node berhasil dimasukkan di bagian awal list...
List: 12 -> !!
Mau membuat node baru? (y/n) y
Masukkan nilai untuk node baru: 90
Membuat node baru...
Berhasil membuat node baru...
Memasukkan node pada bagian awal list...
Node berhasil dimasukkan di bagian awal list...
List: 90 -> 12 -> !!
Mau membuat node baru? (y/n) n
```

Gambar 5.1.b Output Linked List Insertion

Berdasarkan gambar 5.1.b yang merupakan output dari linked list insertion, dapat dilihat bahwa pertama program meminta user untuk memasukkan nilai untuk node pertama dari linked list. Ini sesuai dengan perintah pada baris ke 20. Pada percobaan ini saya memasukkan nilai 12 sebagai node awal (head). Kemudian program menampilkan pesan “Membuat node baru...”. Pada saat ini program sedang menjalankan fungsi `create_new_node`. Kemudian ketika fungsi berhasil dijalankan, program menampilkan pesan “Berhasil membuat node baru...”, “Memasukkan node pada bagian awal list...”, dan “Node berhasil dimasukkan di bagian awal list...”. Kemudian program menjalankan fungsi `display` untuk menampilkan linked list yang sudah dibuat. Pada percobaan kali ini, saya mengisikan y untuk membuat node baru. Kemudian program kembali meminta user untuk memasukkan nilai untuk node yang baru. Kemudian program kembali memasukkan node tersebut di bagian awal list. Kemudian program kembali menampilkan linked list yang terbuat. Kemudian program menanyakan apakah kita ingin membuat node baru. Pada percobaan kali ini, saya mengisikan nilai n untuk tidak membuat node baru. Kemudian program pun selesai.

## 5.2 Percobaan 4-2: Linked List End Insertion

### 5.2.a Source code

```
1  #include <iostream>
2  using namespace std;
3
4  struct node
5  {
6      int info;
7      node *next;
8  } * start, *newptr, *save, *ptr, *rear;
9  node *create_new_node(int);
10 void insert_in_end(node *);
11 void display(node *);
12 int main()
13 {
14     start = rear = NULL;
15     int inf;
16     char ch = 'y';
17     while (ch == 'y' || ch == 'Y')
18     {
19         cout << "Masukkan nilai untuk node baru: ";
20         cin >> inf;
21         cout << "Membuat node baru..." << endl;
22         newptr = create_new_node(inf);
23         if (newptr != NULL)
24         {
25             cout << "Berhasil membuat node baru..." << endl;
26         }
27         else
28         {
29             cout << "Maaf, tidak dapat membuat node baru";
30             return 0;
31         }
32         cout << "Memasukkan node pada bagian akhir list..." << endl;
33         insert_in_end(newptr);
34         cout << "Node berhasil dimasukkan di bagian akhir list..." << endl;
35         cout << "List: ";
36         display(start);
37         cout << "Mau membuat node baru? (y/n) ";
38         cin >> ch;
39     }
40     return 0;
41 }
42 node *create_new_node(int n)
43 {
44     ptr = new node;
45     ptr->info = n;
46     ptr->next = NULL;
47     return ptr;
48 }
49 void insert_in_end(node *np)
50 {
51     if (start == NULL)
52     {
53         start = rear = np;
54     }
55     else
56     {
57         rear->next = np;
58         rear = np;
59     }
60 }
61 void display(node *np)
62 {
63     while (np != NULL)
64     {
65         cout << np->info << " -> ";
66         np = np->next;
67     }
68     cout << "!!!\n";
69 }
```

Gambar 5.2.a Source Code Linked List End Insertion

Berdasarkan gambar 5.2.a yang merupakan source code dari program linked list end insertion, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 4 terdapat deklarasi structure dengan nama node. Di dalamnya, terdapat dua field, yaitu integer info pada baris ke 6 dan node \*next pada baris ke 7. Selain itu terdapat penggunaan start, newptr, save, dan ptr di dalam structure node. Kemudian terdapat inisialisasi create\_new\_node dengan tipe data integer pada baris ke 9. Kemudian pada baris ke 10 terdapat inisialisasi void function insert\_in\_end dengan parameter node. Pada baris ke 11 terdapat deklarasi void function display dengan parameter node. Kemudian pada baris ke 12 terdapat deklarasi fungsi utama. Pada baris ke 14 terdapat inisialisasi nilai start = NULL. Pada baris ke 15 terdapat deklarasi variabel integer dengan nama inf. Pada baris ke 16 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 17 terdapat perulangan while dengan kondisi ch == y atau ch == Y. Di dalam perulangan tersebut, pada baris ke 19 terdapat perintah untuk menampilkan pesan “Masukkan nilai untuk node baru: “. Pada baris ke 20 terdapat perintah agar user memasukkan nilai ke variabel inf. Pada baris ke 21 terdapat perintah untuk menampilkan pesan “Membuat node baru”. Pada baris ke 22 terdapat inisialisasi nilai newptr = nilai dari fungsi create\_new\_node dengan argumen inf. Pada baris ke 23 terdapat percabangan if dengan kondisi jika newptr != NULL, maka tampilkan pesan “Berhasil membuat node baru”. Jika tidak terpenuhi, maka tampilkan pesan “Maaf, tidak dapat membuat node baru” dan program berakhir dengan return 0. Pada baris ke 32 terdapat perintah untuk menampilkan pesan “Memasukkan node pada bagian akhir list...” Pada baris ke 33 terdapat perintah untuk menjalankan fungsi insert\_in\_end dengan argument nilai dari newptr. Pada baris ke 34, terdapat perintah untuk menampilkan pesan “Node berhasil dimasukkan di bagian akhir list...”. Pada baris ke 35 terdapat perintah untuk menampilkan pesan “List: “. Pada baris ke 36 terdapat perintah untuk menjalankan fungsi display dengan argument start. Pada baris ke 37, terdapat perintah untuk menampilkan pesan “Mau membuat node baru?” Kemudian pada baris ke 38 terdapat perintah agar user memasukkan nilai ke dalam variabel ch. Pada baris ke 40 terdapat return 0 yang



menyatakan hasil keluaran dari fungsi main() bahwa program berakhir dengan normal. Pada baris ke 42 terdapat inisialisasi dari node create\_new\_node dengan parameter int n. Pada baris ke 44 terdapat inisialisasi nilai ptr dengan nilai dari new node. Pada baris ke 45 terdapat assignment nilai ptr info dengan n. Pada baris ke 46 terdapat assignment nilai ptr next dengan NULL. Fungsi kemudian mengembalikan nilai ptr. Pada baris ke 49 terdapat inisialisasi fungsi insert\_in\_end dengan parameter node np. Pada baris ke 51, terdapat percabangan if dengan kondisi jika start = Null maka inisialisasi nilai start = rear = np, jika tidak terpenuhi, maka inisialisasi nilai rear next dengan np dan rear dengan np. Pada baris ke 61 terdapat inisialisasi fungsi display dengan parameter node np. Pada baris ke 63 terdapat perulangan while dengan kondisi np != NULL), maka program akan menampilkan nilai dari np info dan kemudian menginisialisasi nilai dari np dengan np next. Kemudian pada baris ke 68 terdapat perintah untuk menampilkan pesan !! dan berpindah baris.

### 5.2.b Output

```
(?) { g++ kode4-3-2.cpp -o kode4-3-2 } ; if (?) { .  
\\kode4-3-2 }  
Masukkan nilai untuk node baru: 12  
Membuat node baru...  
Berhasil membuat node baru...  
Memasukkan node pada bagian akhir list...  
Node berhasil dimasukkan di bagian akhir list...  
List: 12 -> !!  
Mau membuat node baru? (y/n) y  
Masukkan nilai untuk node baru: 90  
Membuat node baru...  
Berhasil membuat node baru...  
Memasukkan node pada bagian akhir list...  
Node berhasil dimasukkan di bagian akhir list...  
List: 12 -> 90 -> !!  
Mau membuat node baru? (y/n) n
```

Gambar 5.2.b Output Linked List End Insertion

Berdasarkan percobaan 5.2.b yang merupakan output dari linked list insertion, dapat dilihat bahwa pertama program meminta user untuk memasukkan nilai untuk node baru. Ini sesuai dengan perintah pada baris ke 20. Pada percobaan kali ini, saya memasukkan nilai 12. Kemudian program menampilkan pesan “Membuat node baru...”. Pada saat ini, program menjalankan fungsi `create_new_ptr` dengan argument `inf`, sesuai dengan perintah pada baris ke 22. Kemudian, jika program berhasil menjalankan fungsi tersebut, program akan menampilkan pesan “Berhasil membuat node baru”. Kemudian, program akan menampilkan pesan “Memasukkan node pada bagian akhir list...” dan menjalankan fungsi `insert_in_end` dengan argument `newptr`. Kemudian program menampilkan pesan “Node berhasil dimasukkan di bagian akhir list..”. Kemudian program akan menampilkan pesan “List: “. Lalu program akan menjalankan fungsi `display` dengan argument `start`, sehingga menampilkan node yang tercipta ke terminal. Pada terminal, terlihat bahwa list yang dihasilkan berbeda dengan program pertama. Ini karena pada program kedua, program memasukkan node baru pada bagian akhir list. Kemudian program menawarkan untuk membuat node baru atau tidak. Pada percobaan ini, saya memilih `y` kembali untuk memasukkan nilai untuk yang kedua kalinya. Kemudian program kembali meminta nilai untuk node baru. Pada percobaan kali ini, saya memasukkan nilai 90. Kemudian program menampilkan pesan “Membuat

node baru...". Pada saat ini, program kembali menjalankan fungsi `create_new_ptr` dengan argument `inf`, sesuai dengan perintah pada baris ke 22. Kemudian, jika program berhasil menjalankan fungsi tersebut, program akan menampilkan pesan "Berhasil membuat node baru". Kemudian, program akan menampilkan pesan "Memasukkan node pada bagian akhir list..." dan menjalankan fungsi `insert_in_end` dengan argument `newptr`. Kemudian program menampilkan pesan "Node berhasil dimasukkan di bagian akhir list..". Kemudian program akan menampilkan pesan "List: ". Lalu program akan menjalankan fungsi `display` dengan argument `start`, sehingga menampilkan node yang tercipta ke terminal. Kemudian program menawarkan untuk membuat node baru atau tidak. Kali ini, saya memilih `n` agar program tidak membuat node baru. Setelah memasukkan nilai `n`, program tidak lagi memenuhi syarat perulangan untuk menambahkan node, sehingga program berhenti pada `return 0`.

### 5.3 Percobaan 4-3: Linked List Delete Node

#### 5.3.a Source Code Linked List Delete Node

```
1 #include <iostream>
2 using namespace std;
3 struct node
4 {
5     int info;
6     node *next;
7 } * start, *newptr, *save, *ptr, *rear;
8 node *create_new_node(int);
9 void insert_node(node *);
10 void display(node *);
11 void delete_node();
12 int main()
13 {
14     start = rear = NULL;
15     int inf;
16     char ch = 'y';
17     while (ch == 'y' || ch == 'Y')
18     {
19         cout << "Masukkan nilai untuk node baru: ";
20         cin >> inf;
21         cout << "Membuat node baru..." << endl;
22         newptr = create_new_node(inf);
23         if (newptr != NULL)
24         {
25             cout << "Berhasil membuat node baru..." << endl;
26         }
27         else
28         {
29             cout << "Maaf, tidak dapat membuat node baru";
30             return 0;
31         }
32         cout << "Memasukkan node pada bagian akhir list..." << endl;
33         insert_node(newptr);
34         cout << "Node berhasil dimasukkan di bagian akhir list..." << endl;
35         cout << "List: ";
36         display(start);
37         cout << "Mau membuat node baru? (y/n) ";
38         cin >> ch;
39     }
40     do
41     {
42         cout << "List:";
43         display(start);
44         cout << "Mau menghapus node pertama? (y/n) ";
45         cin >> ch;
46         if (ch == 'y' || ch == 'Y')
47         {
48             delete_node();
49         }
50     } while (ch == 'y' || ch == 'Y');
51     return 0;
52 }
53 node *create_new_node(int n)
54 {
55     ptr = new node;
56     ptr->info = n;
57     ptr->next = NULL;
58     return ptr;
59 }
60 void insert_node(node *np)
61 {
62     if (start == NULL)
63     {
64         start = rear = np;
65     }
66     else
67     {
68         rear->next = np;
69         rear = np;
70     }
71 }
72 void delete_node()
73 {
74     if (start == NULL)
75     {
76         cout << "Underflow...!!" << endl;
77     }
78     else
79     {
80         ptr = start;
81         start = start->next;
82     }
83 }
84 void display(node *np)
85 {
86     while (np != NULL)
87     {
88         cout << np->info << " -> ";
89         np = np->next;
90     }
91     cout << "!!\n";
92 }
```

Gambar 5.3.a Linked List Delete Node

Berdasarkan gambar 5.3.a yang merupakan source code dari program linked list delete node, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 3 terdapat deklarasi structure dengan nama node. Di dalamnya, terdapat dua field, yaitu integer info pada baris ke 5 dan node \*next pada baris ke 6. Selain itu terdapat penggunaan start, newptr, save, dan ptr di dalam structure node. Kemudian terdapat inisialisasi create\_new\_node dengan tipe data integer pada baris ke 8. Kemudian pada baris ke 9 terdapat inisialisasi void function insert\_node dengan parameter node. Pada baris ke 10 terdapat deklarasi void function display dengan parameter node. Pada baris ke 11 terdapat deklarasi void function delete\_node. Kemudian pada baris ke 12 terdapat deklarasi fungsi utama. Pada baris ke 14 terdapat inisialisasi nilai start = NULL. Pada baris ke 15 terdapat deklarasi variabel integer dengan nama inf. Pada baris ke 16 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 17 terdapat perulangan while dengan kondisi ch == y atau ch == Y. Di dalam perulangan tersebut, pada baris ke 19 terdapat perintah untuk menampilkan pesan “Masukkan nilai untuk node baru: “. Pada baris ke 20 terdapat perintah agar user memasukkan nilai ke variabel inf. Pada baris ke 21 terdapat perintah untuk menampilkan pesan “Membuat node baru”. Pada baris ke 22 terdapat inisialisasi nilai newptr = nilai dari fungsi create\_new\_node dengan argumen inf. Pada baris ke 23 terdapat percabangan if dengan kondisi jika newptr != NULL, maka tampilkan pesan “Berhasil membuat node baru”. Jika tidak terpenuhi, maka tampilkan pesan “Maaf, tidak dapat membuat node baru” dan program berakhir dengan return 0. Pada baris ke 32 terdapat perintah untuk menampilkan pesan “Memasukkan node pada bagian akhir list...” Pada baris ke 33 terdapat perintah untuk menjalankan fungsi insert\_in\_end dengan argument nilai dari newptr. Pada baris ke 34, terdapat perintah untuk menampilkan pesan “Node berhasil dimasukkan di bagian akhir list...”. Pada baris ke 35 terdapat perintah untuk menampilkan pesan “List: “. Pada baris ke 36 terdapat perintah untuk menjalankan fungsi display dengan argument start. Pada baris ke 37, terdapat perintah untuk menampilkan pesan “Mau membuat node baru?” Kemudian pada baris ke 38 terdapat perintah agar user memasukkan nilai ke dalam variabel ch. Pada

baris ke 40, terdapat do dari perulangan while-do. Pada baris ke 42, terdapat perintah untuk menampilkan pesan “List: “. Kemudian pada baris ke 43, terdapat perintah untuk menjalankan perintah display dengan argument start. Pada baris ke 44 terdapat perintah untuk menampilkan pesan “Mau menghapus node pertama? (y/n)” Kemudian terdapat perintah agar user memasukkan nilai y atau n yang nantinya nilai tersebut akan dimasukkan ke dalam variabel ch. Kemudian pada baris ke 46 terdapat percabangan dengan kondisi jika ch = y atau Y, maka jalankan fungsi delete\_node. Kemudian pada baris ke 50 terdapat pendefinisian kondisi while dari perulangan while-do sebelumnya, yaitu jika nilai ch = y atau Y. Pada baris ke 51 terdapat return 0 yang menyatakan hasil keluaran dari fungsi main() bahwa program berakhir dengan normal. Pada baris ke 53 terdapat inisialisasi dari node create\_new\_node dengan parameter int n. Pada baris ke 55 terdapat inisialisasi nilai ptr dengan nilai dari new node. Pada baris ke 56 terdapat assignment nilai ptr info dengan n. Pada baris ke 57 terdapat assignment nilai ptr nextde dengan NULL. Fungsi kemudian mengembalikan nilai ptr. Pada baris ke 60 terdapat inisialisasi fungsi insert\_node dengan parameter node np. Pada baris ke 62, terdapat percabangan if dengan kondisi jika start = Null maka inisialisasi nilai start = rear = np, jika tidak terpenuhi, maka inisialisasi nilai rear next dengan np dan rear dengan np. Pada baris ke 72 terdapat inisialisasi dari fungsi delete\_node. Pada baris ke 74, di dalam fungsi tersebut, terdapat percabangan dengan kondisi jika start = NULL maka tampilkan pesan “Underflow...!!”; jika kondisi tidak terpenuhi, maka inisialisasi nilai ptr dengan start dan nilai start dengan nilai dari start next. Kemudian pada baris ke 84 terdapat inisialisasi fungsi display dengan parameter node np. Pada baris ke 86 terdapat perulangan while dengan kondisi np != NULL), maka program akan menampilkan nilai dari np info dan kemudian menginisialisasi nilai dari np dengan np next. Kemudian pada baris ke 91 terdapat perintah untuk menampilkan pesan !! dan berpindah baris.

### 5.3.b Output Linked List Delete Node

```
++ kode4-3-3.cpp -o kode4-3-3 } ; if ($?) { .\kode4-3-3 }  
-3 }  
Masukkan nilai untuk node baru: 12  
Membuat node baru...  
Berhasil membuat node baru...  
Memasukkan node pada bagian akhir list...  
Node berhasil dimasukkan di bagian akhir list...  
List: 12 -> !!  
Mau membuat node baru? (y/n) y  
Masukkan nilai untuk node baru: 90  
Membuat node baru...  
Berhasil membuat node baru...  
Memasukkan node pada bagian akhir list...  
Node berhasil dimasukkan di bagian akhir list...  
List: 12 -> 90 -> !!  
Mau membuat node baru? (y/n) n  
List:12 -> 90 -> !!  
Mau menghapus node pertama? (y/n) y  
List:90 -> !!  
Mau menghapus node pertama? (y/n) n
```

Gambar 5.3.b Output Linked List Delete Node

Berdasarkan percobaan 5.3.b yang merupakan output dari linked list delete node, dapat dilihat bahwa pertama program meminta user untuk memasukkan nilai untuk node baru. Ini sesuai dengan perintah pada baris ke 20. Pada percobaan kali ini, saya memasukkan nilai 12. Kemudian program menampilkan pesan “Membuat node baru...”. Pada saat ini, program menjalankan fungsi `create_new_ptr` dengan argument `inf`, sesuai dengan perintah pada baris ke 22. Kemudian, jika program berhasil menjalankan fungsi tersebut, program akan menampilkan pesan “Berhasil membuat node baru”. Kemudian, program akan menampilkan pesan “Memasukkan node pada bagian akhir list...” dan menjalankan fungsi `insert_in_end` dengan argument `newptr`. Kemudian program menampilkan pesan “Node berhasil dimasukkan di bagian akhir list..”. Kemudian program akan menampilkan pesan “List: “. Lalu program akan menjalankan fungsi `display` dengan argument `start`, sehingga menampilkan node yang tercipta ke terminal. Kemudian program menawarkan untuk membuat node baru atau tidak. Pada percobaan ini, saya memilih `y` kembali untuk memasukkan nilai untuk yang kedua kalinya. Kemudian program kembali meminta nilai untuk node baru. Pada percobaan kali ini, saya memasukkan nilai 90. Kemudian program menampilkan pesan “Membuat node baru...”. Pada saat ini, program kembali menjalankan fungsi `create_new_ptr`

dengan argument inf, sesuai dengan perintah pada baris ke 22. Kemudian, jika program berhasil menjalankan fungsi tersebut, program akan menampilkan pesan “Berhasil membuat node baru”. Kemudian, program akan menampilkan pesan “Memasukkan node pada bagian akhir list...” dan menjalankan fungsi insert\_in\_end dengan argument newptr. Kemudian program menampilkan pesan “Node berhasil dimasukkan di bagian akhir list..”. Kemudian program akan menampilkan pesan “List: “. Lalu program akan menjalankan fungsi display dengan argument start, sehingga menampilkan node yang tercipta ke terminal. Kemudian program menawarkan untuk membuat node baru atau tidak. Kali ini, saya memilih n agar program tidak membuat node baru. Setelah memasukkan nilai n, program tidak lagi memenuhi syarat perulangan untuk menambahkan node, sehingga program tidak lagi meminta untuk memasukkan nilai untuk node baru. Kemudian program menjalankan fungsi display dengan argument start sehingga menampilkan list dari node yang telah terbuat. Kemudian program menanyakan user apakah ingin menghapus node pertama dari list. Kemudian user diminta untuk memasukkan y atau n. Input dari user nantinya akan menjadi nilai dari variabel ch yang nantinya akan menentukan apakah program harus menghapus node yang pertama atau tidak. Pada percobaan ini, saya memasukkan nilai y untuk menghapus node pertama. Setelah saya menekan y, program menjalankan fungsi delete\_node yang akhirnya menghapus node pertama. Kemudian program kembali menampilkan list yang sudah diperbaharui dan menyisakan nilai 90 sebagai node pertama. Program kembali menanyakan user apakah ingin menghapus node pertama atau tidak. Kali ini, saya memilih n agar program tidak membuat node baru. Setelah memasukkan nilai n, program tidak lagi memenuhi syarat perulangan untuk menambahkan node, sehingga program berhenti pada return 0.



## 5.4 Percobaan 4-4: Linked List Traversal

### 5.4.a Source code Linked List Traversal

```
1 #include <iostream>
2 using namespace std;
3 struct node
4 {
5     int info;
6     node *next;
7 } *start, *newptr, *save, *ptr, *rear;
8 node *create_new_node(int);
9 void insert_node(node *);
10 void travers(node *);
11 int main()
12 {
13     start = rear = NULL;
14     int inf;
15     char ch = 'y';
16     while (ch == 'y' || ch == 'Y')
17     {
18         cout << "Masukkan nilai untuk node baru: ";
19         cin >> inf;
20         cout << "Membuat node baru..." << endl;
21         newptr = create_new_node(inf);
22         if (newptr != NULL)
23         {
24             cout << "Berhasil membuat node baru..." << endl;
25         }
26         else
27         {
28             cout << "Maaf, tidak dapat membuat node baru";
29             return 0;
30         }
31         cout << "Memasukkan node pada bagian akhir list..." << endl;
32         insert_node(newptr);
33         cout << "Node berhasil dimasukkan di bagian akhir list..." << endl;
34         cout << "Mau membuat node baru? (y/n) ";
35         cin >> ch;
36     }
37     cout << "List: ";
38     travers(start);
39     if (start != NULL)
40     {
41         cout << "Mengakses nilai pada satu node: ";
42         cout << start->info << endl;
43         cout << "Alamat dari node ini: ";
44         cout << &start << endl;
45         cout << "Alamat dari node selanjutnya: ";
46         cout << start->next << endl;
47     }
48     if (start->next != NULL)
49     {
50         cout << "Mengakses nilai pada node selanjutnya: ";
51         cout << start->next->info << endl;
52         cout << "Alamat dari node ini: ";
53         cout << start->next << endl;
54         cout << "Alamat dari node selanjutnya: ";
55         cout << start->next->next << endl;
56     }
57     return 0;
58 }
59 node *create_new_node(int n)
60 {
61     ptr = new node;
62     ptr->info = n;
63     ptr->next = NULL;
64     return ptr;
65 }
66 void insert_node(node *np)
67 {
68     if (start == NULL)
69     {
70         start = rear = np;
71     }
72     else
73     {
74         rear->next = np;
75         rear = np;
76     }
77 }
78 void travers(node *np)
79 {
80     while (np != NULL)
81     {
82         cout << np->info << " -> ";
83         np = np->next;
84     }
85     cout << " !!\n";
86 }
```

Gambar 5.4.a Source Code Linked List Traversal

Berdasarkan gambar 5.4.a yang merupakan source code dari program linked list

traversal, dapat dilihat pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 3 terdapat deklarasi structure dengan nama node. Di dalamnya, terdapat dua field, yaitu integer info pada baris ke 5 dan node \*next pada baris ke 6. Selain itu terdapat penggunaan start, newptr, save, dan ptr di dalam structure node. Kemudian terdapat inisialisasi create\_new\_node dengan tipe data integer pada baris ke 8. Kemudian pada baris ke 9 terdapat inisialisasi void function insert\_node dengan parameter node. Pada baris ke 10 terdapat deklarasi void function travers dengan parameter node. Kemudian pada baris ke 11 terdapat deklarasi fungsi utama. Pada baris ke 13 terdapat inisialisasi nilai start = rear = NULL. Pada baris ke 14 terdapat deklarasi variabel integer dengan nama inf. Pada baris ke 15 terdapat inisialisasi variabel char ch dengan nilai y. Pada baris ke 16 terdapat perulangan while dengan kondisi ch == y atau ch == Y. Di dalam perulangan tersebut, pada baris ke 18 terdapat perintah untuk menampilkan pesan “Masukkan nilai untuk node baru: “. Pada baris ke 19 terdapat perintah agar user memasukkan nilai ke variabel inf. Pada baris ke 20 terdapat perintah untuk menampilkan pesan “Membuat node baru”. Pada baris ke 21 terdapat inisialisasi nilai newptr = nilai dari fungsi create\_new\_node dengan argumen inf. Pada baris ke 22 terdapat percabangan if dengan kondisi jika newptr != NULL, maka tampilkan pesan “Berhasil membuat node baru”. Jika tidak terpenuhi, maka tampilkan pesan “Maaf, tidak dapat membuat node baru” dan program berakhir dengan return 0. Pada baris ke 31 terdapat perintah untuk menampilkan pesan “Memasukkan node pada bagian akhir list...”. Pada baris ke 32 terdapat perintah untuk menjalankan fungsi insert\_node dengan argument nilai dari newptr. Pada baris ke 33, terdapat perintah untuk menampilkan pesan “Node berhasil dimasukkan di bagian akhir list...”. Pada baris ke 34 terdapat, perintah untuk menampilkan pesan “Mau membuat node baru?” Kemudian pada baris ke 35 terdapat perintah agar user memasukkan nilai ke dalam variabel ch. Nilai dari variabel ch ini nantinya menentukan apakah program akan membuat node baru atau tidak. Pada baris ke 37 terdapat perintah untuk menampilkan pesan “List: “ dan pada baris ke 38 terdapat perintah untuk menjalankan fungsi travers dengan

argument start. Kemudian pada baris ke 39 terdapat percabangan dengan kondisi jika start != NULL maka program akan menjalankan perintah pada baris ke 41 hingga 46. Pada baris ke 41, terdapat perintah untuk menampilkan pesan “Mengakses nilai pada satu node: “. Kemudian pada baris ke 42 terdapat perintah untuk menampilkan nilai dari start info. Kemudian pada baris ke 43 terdapat perintah untuk menampilkan pesan “Alamat dari node ini: “ dan pada baris ke 44 terdapat perintah untuk menampilkan nilai dari address start. Pada baris ke 45 terdapat perintah untuk menampilkan pesan “Alamat dari node selanjutnya: “ dan pada baris ke 46 terdapat perintah untuk menampilkan nilai dari start next. Kemudian pada baris ke 48 terdapat percabangan if dengan kondisi jika start next != NULL, maka program akan menjalankan perintah pada baris ke 50 hingga 55. Pada baris ke 50 terdapat perintah untuk menampilkan pesan “Mengakses nilai pada node selanjutnya: “. Pada baris ke 51 terdapat perintah untuk menampilkan nilai dari start -> next -> info. Kemudian pada baris ke 52 terdapat perintah untuk menampilkan pesan “Alamat dari node ini: “. Pada baris ke 53 terdapat perintah untuk menampilkan nilai dari start next. Pada baris ke 54 terdapat perintah untuk menampilkan pesan “Alamat dari node selanjutnya: “. Pada baris ke 55 terdapat perintah untuk menampilkan nilai dari start -> next -> next. Kemudian pada baris ke 57 terdapat return 0 yang menyatakan hasil keluaran dari fungsi main() bahwa program berakhir dengan normal. Pada baris ke 59 terdapat inisialisasi dari node create\_new\_node dengan parameter int n. Pada baris ke 61 terdapat inisialisasi nilai ptr dengan nilai dari new node. Pada baris ke 62 terdapat assignment nilai ptr info dengan n. Pada baris ke 63 terdapat assignment nilai ptr next dengan NULL. Fungsi kemudian mengembalikan nilai ptr. Pada baris ke 66 terdapat inisialisasi fungsi insert\_node dengan parameter node np. Pada baris ke 68, terdapat percabangan if dengan kondisi jika start = Null maka inisialisasi nilai start = rear = np, jika tidak terpenuhi, maka inisialisasi nilai rear next dengan np dan rear dengan np. Pada baris ke 78 terdapat inisialisasi fungsi travers dengan parameter node np. Pada baris ke 80 terdapat perulangan while dengan kondisi jika np != NULL, maka program akan menampilkan nilai dari np info dan kemudian menginisialisasi nilai dari np dengan np next. Kemudian pada baris ke 68 terdapat perintah untuk menampilkan pesan !! dan berpindah baris.

#### 5.4.b Output Linked List Traversal

```
(?) { g++ kode4-3-4.cpp -o kode4-3-4 } ; if (?) {  
 \kode4-3-4 }  
Masukkan nilai untuk node baru: 12  
Membuat node baru...  
Berhasil membuat node baru...  
Memasukkan node pada bagian akhir list...  
Node berhasil dimasukkan di bagian akhir list...  
Mau membuat node baru? (y/n) y  
Masukkan nilai untuk node baru: 90  
Membuat node baru...  
Berhasil membuat node baru...  
Memasukkan node pada bagian akhir list...  
Node berhasil dimasukkan di bagian akhir list...  
Mau membuat node baru? (y/n) n  
List: 12 -> 90 -> !!  
Mengakses nilai pada satu node: 12  
Alamat dari node ini: 0x408030  
Alamat dari node selanjutnya: 0x1096e20  
Mengakses nilai pada node selanjutnya: 90  
Alamat dari node ini: 0x1096e20  
Alamat dari node selanjutnya: 0
```

Gambar 5.4.b Output Linked List Traversal

Berdasarkan gambar 5.4.b yang merupakan gambar dari output linked list traversal, dapat dilihat bahwa pertama program meminta user untuk memasukkan nilai untuk node pertama dari linked list. Ini sesuai dengan perintah pada baris ke 19. Pada percobaan ini saya memasukkan nilai 12 sebagai node awal (head). Kemudian program menampilkan pesan “Membuat node baru...”. Pada saat ini program sedang menjalankan fungsi `create_new_node`. Kemudian ketika fungsi berhasil dijalankan, program menampilkan pesan “Berhasil membuat node baru...”, “Memasukkan node pada bagian akhir list...”, dan “Node berhasil dimasukkan di bagian akhir list...”. Kemudian program kembali menanyakan kepada user apakah ingin membuat node baru. Pada percobaan kali ini, saya mengisikan y untuk membuat node baru. Kemudian program kembali meminta user untuk memasukkan nilai untuk node yang baru. Kemudian program kembali memasukkan node tersebut di bagian akhir list. Kemudian program kembali menanyakan apakah kita ingin membuat node baru. Pada percobaan kali ini, saya mengisikan nilai n untuk tidak membuat node baru. Kemudian program menjalankan fungsi `travers` sehingga menampilkan list yang terbentuk, sesuai dengan perintah pada baris ke 37 dan 38. Kemudian program menampilkan pesan mengakses nilai pada satu node,

menampilkan nilai dari start->info, yaitu 12, menampilkan pesan “Alamat dari node ini”, menampilkan address dari start, yaitu 0x408030, menampilkan pesan “alamat dari node selanjutnya”, dan menampilkan nilai dari start->next”, yaitu 0x1096e20. Ini dijalankan karena program memenuhi kondisi untuk percabangan pada baris ke 39. Kemudian program kembali menampilkan pesan “Mengakses nilai pada node selanjutnya”, menampilkan nilai dari start->next->info, yaitu 90, menampilkan pesan “Alamat dari node ini”, menampilkan nilai dari start->next, yaitu 0x1096e20, menampilkan pesan “Alamat dari node selanjutnya: “, dan menampilkan nilai dari start->next->next, yaitu 0. Ini dijalankan karena kondisi pada percabangan baris ke 48 terpenuhi. Untuk nilai dari start->next->next adalah 0 karena list hanya memiliki dua node. Jika list memiliki lebih dari satu node, maka program akan menampilkan alamat dari node berikutnya. Namun jika terdapat lebih dari dua node, maka program tetap hanya akan menampilkan detail dari node pertama dan kedua karena pada program hanya terdapat dua kali pembacaan dari node.

## VI. KESIMPULAN

Adapun kesimpulan dari percobaan ini adalah sebagai berikut :

1. Berdasarkan percobaan 5.1 Linked List Insertion, dapat disimpulkan bahwa setiap node baru yang dibuat akan menjadi head dari node tersebut karena node ditambahkan di bagian awal list.
2. Berdasarkan percobaan 5.2 Linked List End Insertion, dapat disimpulkan bahwa node yang ditambahkan di bagian akhir tidak merubah head dari link tersebut.
3. Berdasarkan percobaan 5.3 Linked List Delete Node, dapat disimpulkan bahwa ketika kita menghapus node pertama, maka head dari node tersebut akan berubah. Selain itu ketika node terakhir telah dihapus dan kita memaksa untuk menghapus suatu node, maka akan terjadi underflow.
4. Berdasarkan percobaan 5.4 Linked List Traversal, dapat disimpulkan bahwa pada single linked list, pada bagian akhir (tail) dari list tersebut, jika kita memeriksa address dari node berikutnya, maka program akan menampilkan angka 0.
5. Berdasarkan percobaan 5.1 Linked List Insertion dan 5.2 Linked List End Insertion, dapat disimpulkan bahwa suatu node dapat ditambahkan pada bagian belakang atau depan dari sebuah linked list.
6. Berdasarkan percobaan 5.1 Linked List Insertion, 5.2 Linked List End Insertion, 5.3 Linked List Delete Node, dan 5.4 Linked List Traversal, dapat disimpulkan bahwa untuk membuat node, diperlukan suatu tipe data khusus, yaitu struct. Kemudian di dalam struct tersebut, kita dapat menuliskan field dan metode-metode yang diperlukan dalam operasi linked list.

## DAFTAR PUSTAKA

- Erliansyah Nasution dan Indra Yatini B.2005. Algoritma dan Struktur Data. Graha Ilmu. Yogyakarta.
- Sjukani, Moh. 2010. (Algoritma Dan Struktur Data I) Dengan C, C++ dan Java. Jakarta: Mitra Wacana Media
- Muhammad, M.A. (018. Struktur Data. Modul Struktur Data Unila PSTI. 2 (2) : 1 - 2.
- Muhammad, Meizano Ardhi. 2018. Modul Praktikum Struktur Data. Lampung Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung.
- Ulum, M. Bahrul Ulum.2018. Modul Praktikum Struktur Data. Jakarta Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Esa Unggul.

## TUGAS AKHIR

### 1. Source code

Adapun source code dari program adalah sebagai berikut

```
1  #include <iostream>
2  using namespace std;
3
4  struct Student
5  {
6      int id;
7      string name;
8      string address;
9      int year;
10     Student *next;
11 } *head, *curr, *newstud;
12
13 void addStudent(int id, string name, string address, int year)
14 {
15     newstud = new Student();
16     newstud->id = id;
17     newstud->name = name;
18     newstud->address = address;
19     newstud->year = year;
20     newstud->next = head;
21     head = newstud;
22 }
23
24 void display()
25 {
26     curr = head;
27     if(curr == NULL)
28     {
29         cout << "No data found" << endl;
30     }
31     else
32     {
33         while(curr != NULL)
34         {
35             cout << "ID : " << curr->id << endl;
36             cout << "Name : " << curr->name << endl;
37             cout << "Address : " << curr->address << endl;
38             cout << "Start studying in : " << curr->year << endl;
39             cout << "Memory address : " << &curr << endl;
40             curr = curr->next;
41         }
42     }
43 }
44
45 int main()
46 {
47     addStudent(13130, "Kang Asep", "New Village", 2021);
48     display();
49 }
```

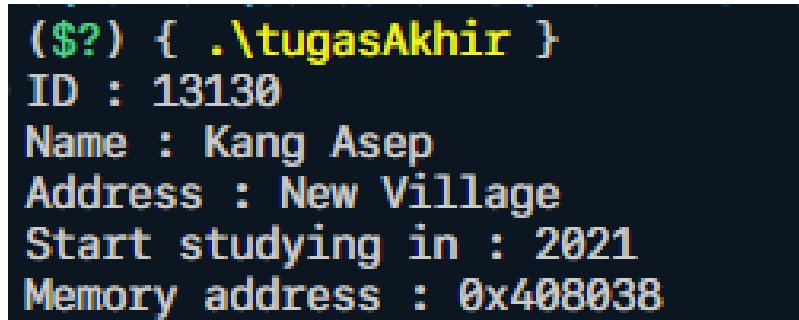
Gambar 1 Source code

Berdasarkan gambar 1 yang merupakan source code dari program, dapat dilihat bahwa pada baris ke 1 terdapat penggunaan library iostream ke dalam program yang



berfungsi untuk menambahkan perintah input dan output. Kemudian pada baris ke 2 terdapat instruksi penggunaan penamaan standar untuk compiler. Kemudian pada baris ke 4 terdapat deklarasi structure dengan nama Student. Pada structure tersebut terdapat 5 field yaitu integer id pada baris ke 6, string name pada baris ke 7, string address pada baris ke 8, integer year pada baris ke 9, dan address dari Student pada baris ke 10. Pada baris ke 11, terdapat pendeklarasian method, yaitu head, curr, dan newstud. Head berfungsi untuk mengambil node pertama. curr berfungsi untuk menunjuk pada suatu node yang sedang dibaca, dan newstud sebagai perantara untuk membuat node baru. Kemudian pada baris ke 13 terdapat inisialisasi void function dengan nama addStudent dengan parameter integer id, string name, string address, dan integer year. Pada baris ke 15, di dalam void function addStudent, terdapat inisialisasi newstud sebagai object dari node Student. Pada baris ke 16, terdapat inisialisasi id pada newstud dengan nilai dari id. Pada baris ke 16, terdapat inisialisasi nilai id pada newstud dengan nilai dari id dari parameter fungsi addStudent. Pada baris ke 17, terdapat inisialisasi nilai name pada newstud dengan nilai dari name dari parameter fungsi addStudent. Pada baris ke 18, terdapat inisialisasi nilai address pada newstud dengan nilai dari address dari parameter fungsi addStudent. Pada baris ke 19, terdapat inisialisasi nilai year pada newstud dengan nilai dari year dari parameter fungsi addStudent. Pada baris ke 20, terdapat inisialisasi nilai next pada newstud dengan nilai dari head. Pada baris ke 24 terdapat inisialisasi void function display. Pada baris ke 26, di dalam void function display, terdapat inisialisasi nilai curr = head. Pada baris ke 27, terdapat percabangan dengan kondisi jika nilai dari curr = NULL maka program akan menjalankan perintah pada baris ke 35 hingga 39, yaitu menampilkan id, name, address, year, dan alamat dari node, serta menginisialisasi nilai curr dengan nilai dari curr->next. Kemudian pada baris ke 45 terdapat deklarasi fungsi integer main. Di dalamnya, pada baris ke 47 terdapat perintah untuk menjalankan fungsi addStudent dengan argument 13130, "Kang Asep", "New Village", dan 2021. Pada baris ke 48 terdapat perintah untuk menjalankan fungsi display.

## 2. Output

A screenshot of a terminal window with a black background and yellow text. The text displays the output of a program, showing student information and a memory address.

```
($?) { .\tugasAkhir }  
ID : 13130  
Name : Kang Asep  
Address : New Village  
Start studying in : 2021  
Memory address : 0x408038
```

Gambar 2 Output program

Berdasarkan gambar 2 yang merupakan output dari program, dapat dilihat bahwa pada baris ke 47, pertama program menjalankan fungsi dari `addStudent`, yaitu menambahkan siswa baru pada node. Kemudian pada baris ke 48, program menjalankan fungsi `display` sehingga program menampilkan id dari siswa tersebut, yaitu 13130. Kemudian program menampilkan nama dari siswa, yaitu Kang Asep. Kemudian program menampilkan alamat dari siswa, yaitu New Village. Kemudian program menampilkan tahun pertama siswa tersebut masuk ke sekolah, yaitu pada 2021. Data ID, nama, alamat, dan tahun awal siswa masuk ke sekolah didapat dari argument pada pemanggilan fungsi `addStudent` pada baris ke 47 pada program. Kemudian program menampilkan alamat dari node tersebut, yaitu 0x408038.