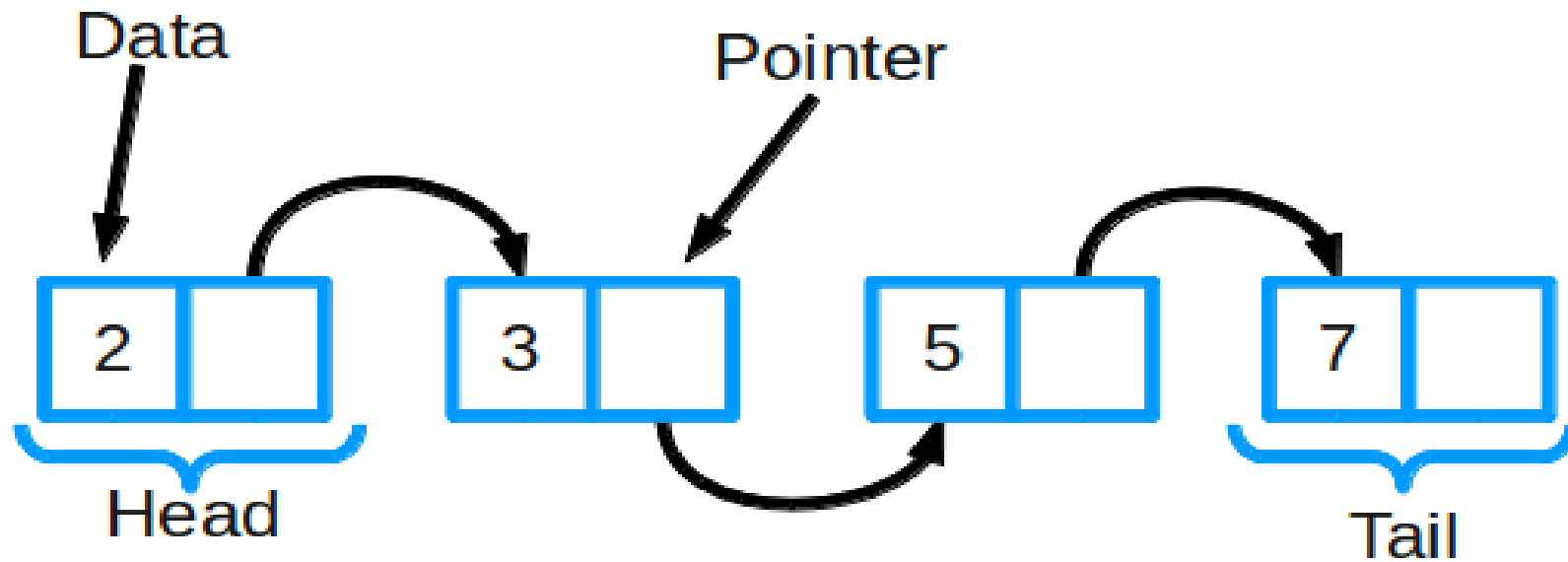
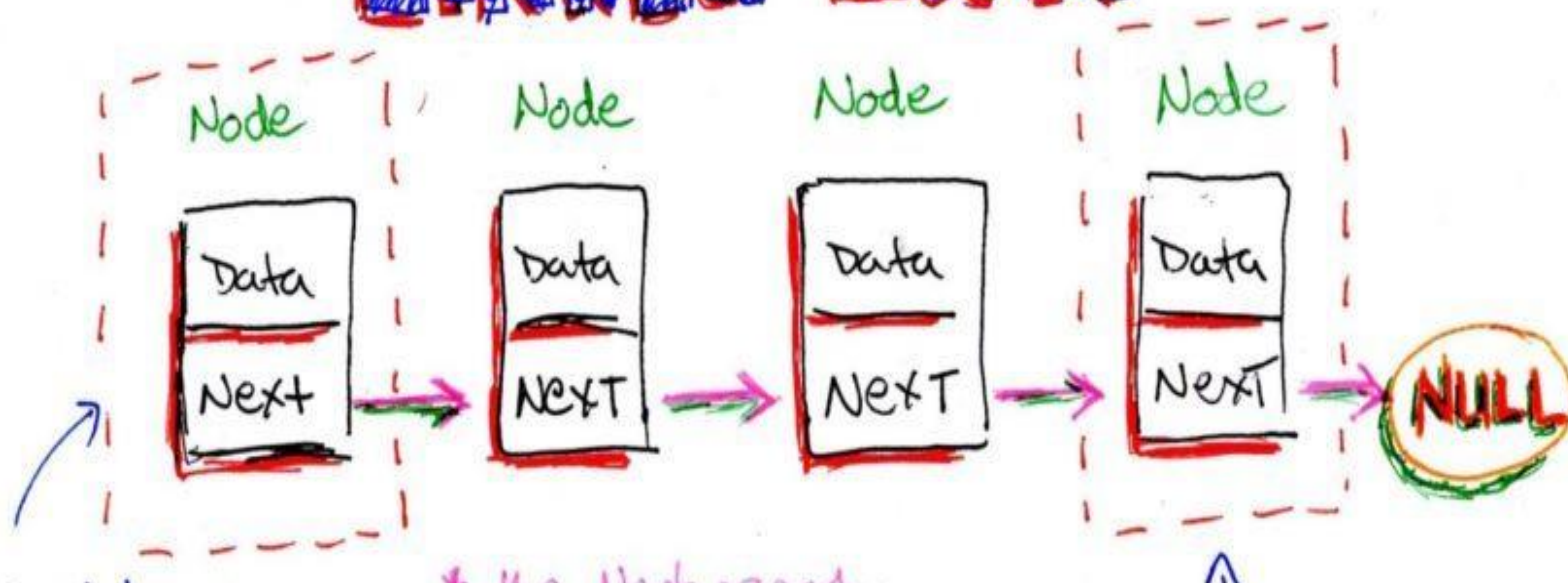


Linked Lists



Struktur Data

Fitri Nuraeni, M.Kom

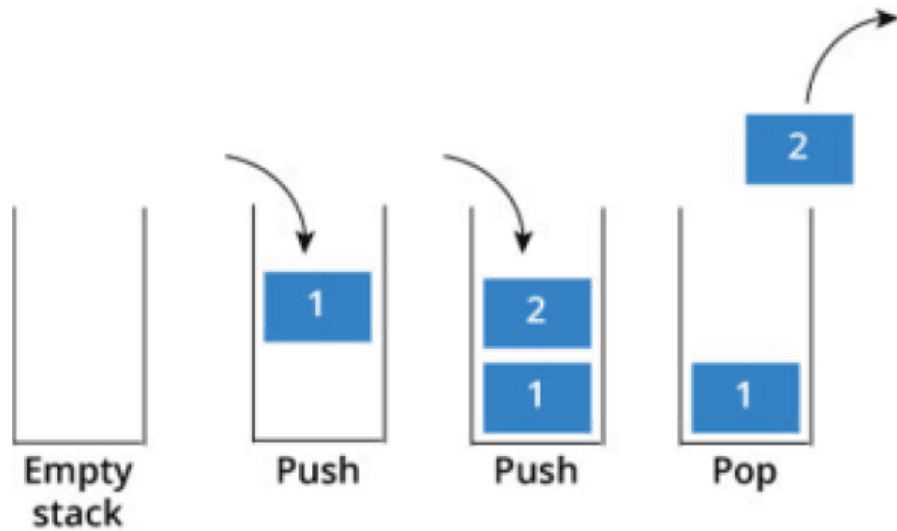
<http://latihan.nufi3.com>

Struktur Data Senarai (List)

Pertemuan Ke-14

Kilas Balik

Stack Vs Queue



Stack



Queue

Program Simulasi Stack & Queue Pada C++

Simulasi Nomor Antrian Customer

```
1 #include <iostream>
2 #define n 10
3 using namespace std;
4
5 int First, Last;
6 int antrian[n];
7
8 void buatKosong(){
9     First = -1;
10    Last = -1;
11 }
12
```

Perhatikan Nomor Barisnya!!!

```
13 bool isKosong(){
14     return (First < 0);
15 }
16
17 bool isPenuh(){
18     return (Last == (n-1));
19 }
20
21 int getDepan(){
22     return First;
23 }
24
25 int getBelakang(){
26     return Last;
27 }
```

Simulasi Nomor Antrian Customer (2)

```
28
29 void tampilkanAntrian(){
30     if(isKosong() == false){
31         cout<<"On Process : "<<antrian[getDepan()]<<"\n";
32         cout<<"Selanjutnya : ";
33         for(int i=getBelakang(); i >= (getDepan()+1); i--){
34             cout<<antrian[i]<<" | ";
35         }
36         cout<<"\n";
37     }else{
38         cout<<"Antrian KOSONG\n";
39     }
40 }
41
42 int selisihAntrian(){
43     int awal = antrian[getDepan()];
44     int akhir = antrian[getBelakang()];
45
46     return akhir - awal;
47 }
```

Perhatikan Nomor Barisnya!!!

Simulasi Nomor Antrian Customer (3)

```
48
49 ▾ int tamuDatang(){
50 ▾     if(isPenuh() == false){
51         int x = antrian[getBelakang()] + 1; //create nomor antrian
52
53 ▾         if(isKosong() == true){
54             First += 1;
55         }
56         Last += 1;
57         antrian[Last] = x;
58
59         cout<<"\n\nNo Antrian : "<<x<<"\n";
60         cout<<"On Process : "<<antrian[getDepan()]<<"\n";
61         cout<<"Anda dilayani setelah "<<selisihAntrian()<<" berikutnya\n\n";
62 ▾     }else{
63         cout<<"\n\nAntrian PENUH | Kapasitas = "<<n;
64         cout<<"On Process : "<<antrian[getDepan()]<<"\n\n";
65     }
66
67     return getBelakang();
68 }
```

Perhatikan Nomor Barisnya!!!

Simulasi Nomor Antrian Customer (4)

```
69
70 int tamuSelesai(){
71     int x = antrian[First];
72     if( isKosong() == false){
73         if(getDepan() == getBelakang()){ //satu elemen di Antrian
74             antrian[First] = 0;
75             buatKosong(); // Antrian jadi KOSONG
76         }else{
77             antrian[First] = 0; // hapus elemen pertama
78             //geser elemen
79             int posisi = First + 1;
80             while(posisi <= getBelakang()){
81                 antrian[posisi-1] = antrian[posisi];
82                 posisi++;
83             }
84             antrian[Last] = 0;
85             Last -= 1;
86         }
87
88         cout<<"\n\nNext Process : "<<antrian[getDepan()]<<"\n";
89         cout<<"Sisa Tamu : "<<selisihAntrian()<<"\n\n";
90     }else{
91         cout<<"\n\nAntrian KOSONG \n";
92     }
93
94     return getBelakang();
95 }
```

Perhatikan Nomor Barisnya!!!

Simulasi Nomor Antrian Customer (5)

```
97 int main(){
98     char Y,menu;
99     buatKosong();
100     tampilkanAntrian();
101
102     Y = 'Y';
103     while(Y == 'Y'){
104         cout<<"\n\n===== \n";
105         cout<<"Pilihan Menu : \n";
106         cout<<"<N = new> Tamu Baru \n";
107         cout<<"<F = finish> Tamu Selesai \n";
108         cout<<"===== \n";
109         cout<<"Pilihan <N/F> : ";
110         cin>>menu;
```

```
111
112     if(menu == 'N'){
113         tamuDatang();
114         tampilkanAntrian();
115     }else if(menu == 'F'){
116         tamuSelesai();
117         tampilkanAntrian();
118     }else{
119         cout<<"Menu tidak valid!!!\n";
120     }
121
122     cout<<"\n===== \n";
123     cout<<"Proses lagi <Y/T>";
124     cin>>Y;
125 }
126
127     return 0;
128 }
```

Perhatikan Nomor Barisnya!!!

Running Program

```
Antrian KOSONG
```

```
=====
Pilihan Menu :
```

```
<N = new> Tamu Baru
```

```
<F = finish> Tamu Selesai
```

```
=====
Pilihan <N/F> : N
```

```
No Antrian : 1
```

```
On Process : 1
```

```
Anda dilayani setelah 0 berikutnya
```

```
On Process : 1
```

```
Selanjutnya :
```

```
=====
Proses lagi <Y/T>Y
```

```
=====
Pilihan Menu :
```

```
<N = new> Tamu Baru
```

```
<F = finish> Tamu Selesai
```

```
=====
Pilihan <N/F> : N
```

```
No Antrian : 2
```

```
On Process : 1
```

```
Anda dilayani setelah 1 berikutnya
```

```
On Process : 1
```

```
Selanjutnya : 2 |
```

```
=====
Proses lagi <Y/T>Y
```

```
=====
Pilihan Menu :
```

```
<N = new> Tamu Baru
```

```
<F = finish> Tamu Selesai
```

```
=====
Pilihan <N/F> : N
```

```
No Antrian : 3
```

```
On Process : 1
```

```
Anda dilayani setelah 2 berikutnya
```

```
On Process : 1
```

```
Selanjutnya : 3 | 2 |
```

```
=====
Proses lagi <Y/T>Y
```

Running Program

```
=====
Pilihan Menu :
<N = new> Tamu Baru
<F = finish> Tamu Selesai
=====
Pilihan <N/F> : F

Next Process : 2
Sisa Tamu : 1

On Process : 2
Selanjutnya : 3 |

=====
Proses lagi <Y/T>Y
```

```
=====
Pilihan Menu :
<N = new> Tamu Baru
<F = finish> Tamu Selesai
=====
Pilihan <N/F> : N

No Antrian : 4
On Process : 2
Anda dilayani setelah 2 berikutnya

On Process : 2
Selanjutnya : 4 | 3 |

=====
Proses lagi <Y/T>T

...Program finished with exit code 0
Press ENTER to exit console.
```

Simulasi Pengelolaan Histori Browser

```
1  #include <iostream>
2  #include <string>
3  #define n 10
4  using namespace std;
5
6  //global variabel
7  int Top, Top2;
8  std::string histori[n];
9  std::string moveon[n];
10 std::string kata;
11
12 void buatKosong(int obj){
13     if(obj == 0)
14         Top = -1;
15     else if(obj == 1)
16         Top2 = -1;
17 }
```

```
18
19 bool isKosong(int obj){
20     if(obj == 0){
21         if(Top < 0){
22             return true;
23         }else{
24             return false;
25         }
26     }else if(obj == 1){
27         if(Top2 < 0){
28             return true;
29         }else{
30             return false;
31         }
32     }else{
33         return false;
34     }
35 }
```

Simulasi Pengelolaan Histori Browser

```
37 bool isPenuh(int obj){
38     if(obj == 0){
39         if(Top == (n-1)){
40             return true;
41         }else{
42             return false;
43         }
44     }else if(obj == 1){
45         if(Top2 == (n-1)){
46             return true;
47         }else{
48             return false;
49         }
50     }else{
51         return false;
52     }
53 }
```

```
55 int getTop(int obj){
56     if(obj == 0)
57         return Top;
58     else
59         return Top2;
60 }
61
62 void tampilkanStack(){
63     if(iskosong(0) == false){
64         cout<<"\nRiwayat Menjelajah : ";
65         for(int i=getTop(0); i>=0; i--){
66             cout<<histori[i]<<" | ";
67         }
68         cout<<"\n\n";
69     }else{
70         cout<<"\nRiwayat KOSONG\n\n";
71     }
72 }
73
74 int jumlahItem(int obj){
75     if(obj == 0)
76         return getTop(0)+1;
77     else
78         return getTop(1)+1;
79 }
80
```

Simulasi Pengelolaan Histori Browser

```
81 int jelajahBaru(){
82     if(isPenuh(0) == false){
83         Top += 1;
84         histori[Top] = kata;
85
86         cout<<"\n\n";
87         cout<<"Membuka jelajah : "<<kata<<"\n";
88         cout<<"Jumlah Riwayat = "<<jumlahItem(0)<<"\n";
89     }else{
90         cout<<"\n\nRiwayat PENUH\n";
91         cout<<"Jumlah Riwayat = "<<jumlahItem(0)<<"\n";
92         cout<<"Kapasitas = "<<n<<"\n\n";
93     }
94
95     return getTop(0);
96 }
97
```

Simulasi Pengelolaan Histori Browser

```
98 ▾ int forwardBaru(){
99 ▾     if(isPenuh(1) == false){
100         Top2 += 1;
101         moveon[Top2] = kata;
102 ▾     }else{
103         cout<<"\n\nForward PENUH\n";
104         cout<<"Jumlah Forward = "<<jumlahItem(1)<<"\n";
105         cout<<"Kapasitas = "<<n<<"\n\n";
106     }
107
108     return getTop(1);
109 }
110
```

Simulasi Pengelolaan Histori Browser

```
111 int jelajahKembali(){
112     if(isKosong(0) == false){
113         kata = histori[Top];
114         forwardBaru();
115
116         histori[Top] = "";
117         Top -= 1;
118
119         cout<<"Kembali jelajah "<<kata<<"\n";
120         cout<<"Jumlah Riwayat = "<<jumlahItem(0)<<"\n";
121     }else{
122         cout<<"\n\nRiwayat KOSONG\n";
123         cout<<"Kapasitas = "<<n<<"\n\n";
124     }
125
126     return getTop(0);
127 }
128
```


Simulasi Pengelolaan Histori Browser

```
129 int jelajahForward(){
130     if(iskosong(1) == false){
131         kata = moveon[Top2];
132         jelajahBaru();
133
134         moveon[Top2] = "";
135         Top2 -= 1;
136     }else{
137         cout<<"\n\nForward KOSONG\n";
138         cout<<"Kapasitas = "<<n<<"\n\n";
139     }
140
141     return getTop(1);
142 }
```

Simulasi Pengelolaan Histori Browser

```
144 int main()
145 {
146     char Y,menu;
147
148     buatKosong(0);
149     buatKosong(1);
150     tampilkanStack();
151
152     Y = 'Y';
153     while(Y == 'Y'){
154         cout<<"\n\n===== \n";
155         cout<<"Pilihan Menu : \n";
156         cout<<"<N = new> Jelajah Baru \n";
157         cout<<"<B = Back> Kembali Ke Halaman Sebelumnya \n";
158         cout<<"<F = Forward> Maju Ke Halaman Sebelumnya \n";
159         cout<<"===== \n";
160         cout<<"Pilihan <N/B/F> : ";
161         cin>>menu;
```

Simulasi Pengelolaan Histori Browser

```
162
163     if(menu == 'N'){
164         cout<<"Masukan URL = ";
165         cin>>kata;
166
167         jelajahBaru();
168         tampilkanStack();
169     }else if(menu == 'B'){
170         jelajahKembali();
171         tampilkanStack();
172     }else if(menu == 'F'){
173         jelajahForward();
174         tampilkanStack();
175     }else{
176         cout<<"Menu tidak valid!!!\n";
177     }
178
179     cout<<"\n===== \n";
180     cout<<"Proses lagi <Y/T>";
181     cin>>Y;
182 }
183
184 return 0;
185 }
```

Running Program

```
Riwayat KOSONG
```

```
=====
```

```
Pilihan Menu :
```

```
<N = new> Jelajah Baru
```

```
<B = Back> Kembali Ke Halaman Sebelumnya
```

```
<F = Forward> Maju Ke Halaman Sebelumnya
```

```
=====
```

```
Pilihan <N/B/F> : N
```

```
Masukan URL = latiha.nufi3.com
```

```
Membuka jelajah : latiha.nufi3.com
```

```
Jumlah Riwayat = 1
```

```
Riwayat Menjelajah : latiha.nufi3.com |
```

```
=====
```

```
Proses lagi <Y/T>Y
```

```
=====
```

```
Pilihan Menu :
```

```
<N = new> Jelajah Baru
```

```
<B = Back> Kembali Ke Halaman Sebelumnya
```

```
<F = Forward> Maju Ke Halaman Sebelumnya
```

```
=====
```

```
Pilihan <N/B/F> : N
```

```
Masukan URL = stmik-tasikmalaya.ac.id
```

```
Membuka jelajah : stmik-tasikmalaya.ac.id
```

```
Jumlah Riwayat = 2
```

```
Riwayat Menjelajah : stmik-tasikmalaya.ac.id | latiha.nufi3.com |
```

```
=====
```

```
Proses lagi <Y/T>Y
```

Running Program

```
=====
Pilihan Menu :
<N = new> Jelajah Baru
<B = Back> Kembali Ke Halaman Sebelumnya
<F = Forward> Maju Ke Halaman Sebelumnya
=====
Pilihan <N/B/F> : N
Masukan URL = nufi3.com

Membuka jelajah : nufi3.com
Jumlah Riwayat = 3

Riwayat Menjelajah : nufi3.com | stmik-tasikmalaya.ac.id | latiha.nufi3.com |

=====
Proses lagi <Y/T>Y
```

Running Program

```
=====
Pilihan Menu :
<N = new> Jelajah Baru
<B = Back> Kembali Ke Halaman Sebelumnya
<F = Forward> Maju Ke Halaman Sebelumnya
=====
Pilihan <N/B/F> : B
Kembali jelajah nufi3.com
Jumlah Riwayat = 2

Riwayat Menjelajah : stmik-tasikmalaya.ac.id | latiha.nufi3.com |

=====
Proses lagi <Y/T>Y
```

Running Program

```
=====
Pilihan Menu :
<N = new> Jelajah Baru
<B = Back> Kembali Ke Halaman Sebelumnya
<F = Forward> Maju Ke Halaman Sebelumnya
=====
Pilihan <N/B/F> : F

Membuka jelajah : nufi3.com
Jumlah Riwayat = 3

Riwayat Menjelajah : nufi3.com | stmik-tasikmalaya.ac.id | latiha.nufi3.com |

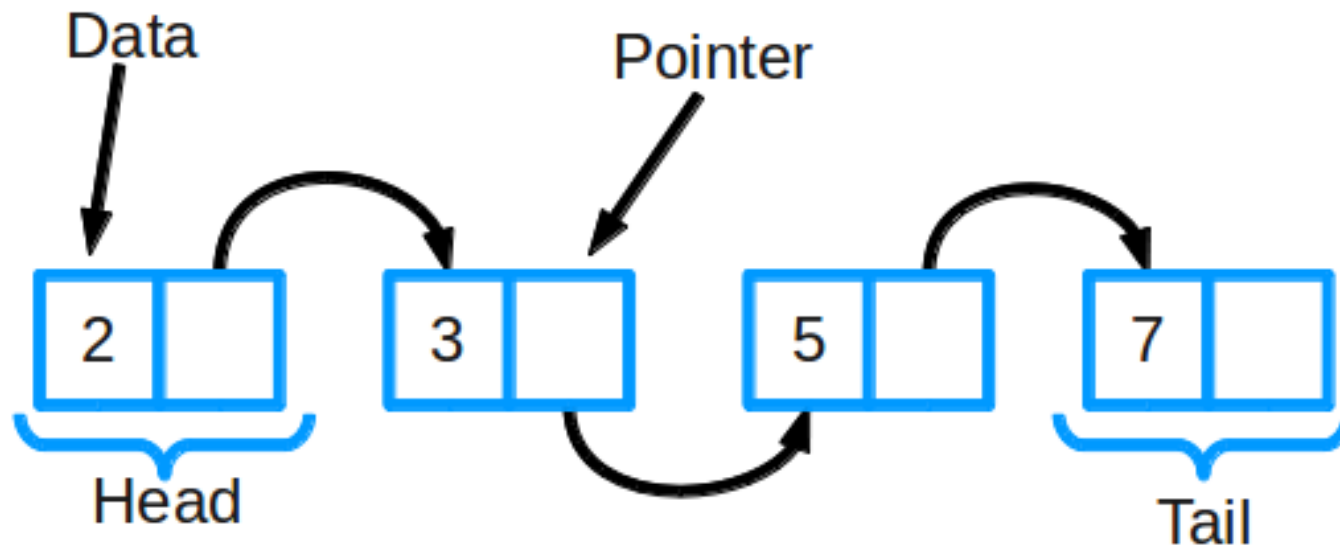
=====
Proses lagi <Y/T>T

...Program finished with exit code 0
Press ENTER to exit console.
```

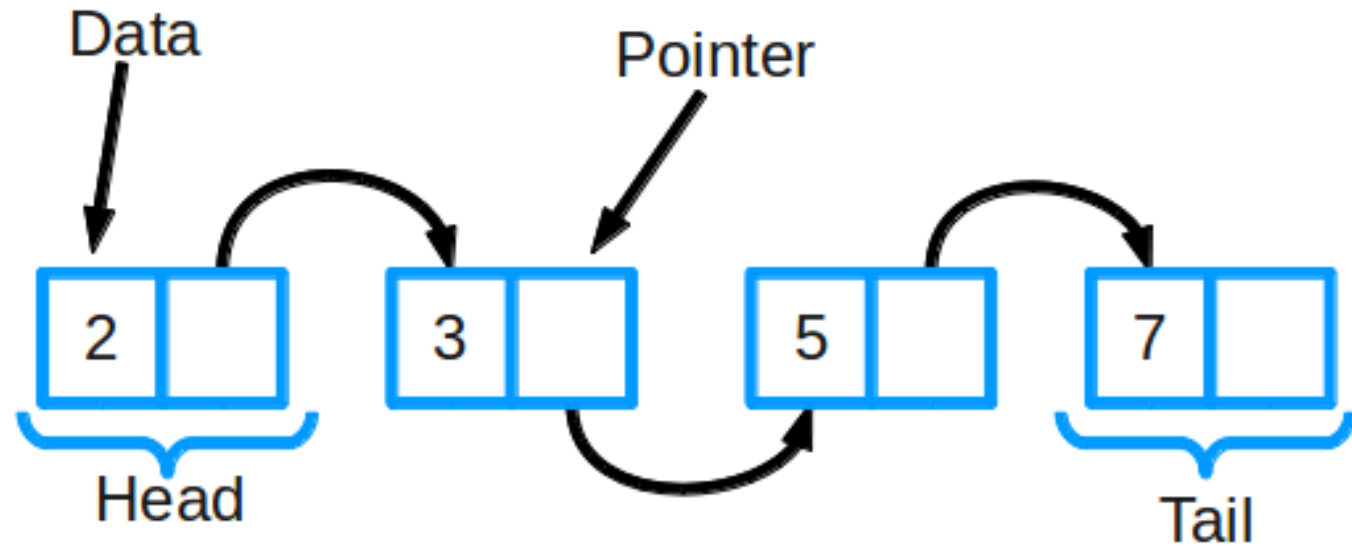
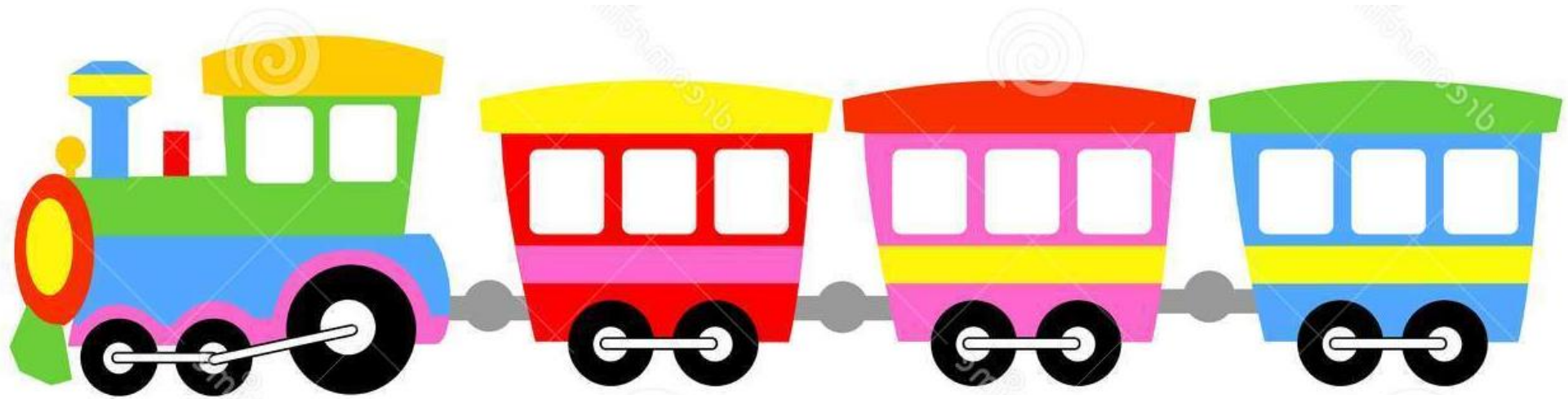
Senarai (List)

Pengenalan List

- Sebuah konsep struktur data yang sangat dasar pada pemrograman agar lebih **fleksibel** :
 - Dimana setiap elemen akan ditambahkan saat dibutuhkan
 - Tidak dialokasikan dengan tempat tertentu dari awal
- Sekumpulan elemen dengan struktur tertentu



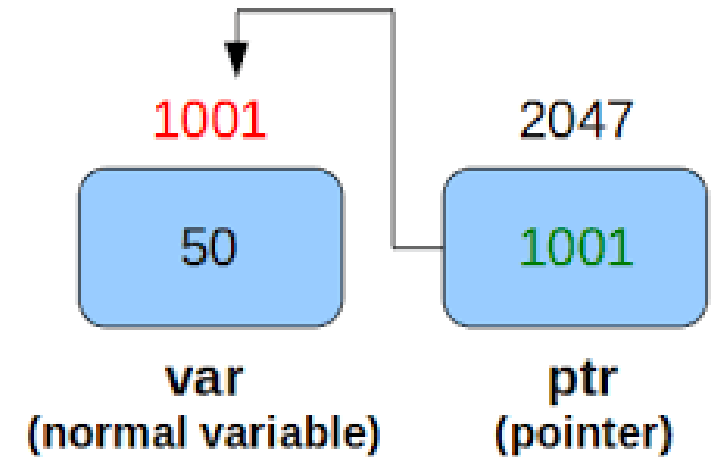
Ilustrasi List



Apa itu **Pointer**???

Pointer

- Pointer adalah sebuah variabel atau object yang menunjuk ke variabel atau obyek lainnya.
- Untuk program C ++, **memori komputer** seperti sukseksi sel-sel memori, masing-masing berukuran satu byte, dan masing-masing **memiliki alamat unik**, sehingga dapat dengan mudah ditemukan dalam memori melalui alamatnya yang unik.
- Ketika suatu variabel dideklarasikan, *memori yang diperlukan untuk menyimpan nilainya ditetapkan lokasi tertentu dalam memori* (alamat memorinya)



Pointer pada C++

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int var = 50;
7
8      cout<<&var<<" memiliki nilai "<<var<<endl;
9
10     return 0;
11 }
```

Tergantung pengalamat di
memori computer masing-
masing



```
0x7ffdc518728c memiliki nilai 50
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

Pointer pada C++

```
main.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int var = 50;
7      int *ptr = &var;
8
9      cout<<&var<<" memiliki nilai "<<var<<endl;
10     cout<<"ptr = "<<ptr<<endl;
11
12     return 0;
13 }
```

```
0x7fffd2edc0b4 memiliki nilai 50
ptr = 0x7fffd2edc0b4

...Program finished with exit code 0
Press ENTER to exit console.
```

Ptr = variabel pointer, menunjuk ke variabel **var** yang bertipe **int**, dengan menggunakan tanda asterisk * (**int * ptr**).

Gunakan operator **&** untuk menyimpan alamat memori dari variabel yang disebut makanan, dan menetapkannya ke pointer.

Sekarang, ptr menyimpan nilai alamat memori var.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int var = 50;
7      int angka = 1;
8      int *ptr = &var;
9
10     cout<<&var<<" memiliki nilai "<<var<<endl;
11     cout<<"ptr = "<<ptr<<endl;
12
13     ptr = &angka;
14     cout<<"ptr = "<<ptr<<endl;
15     cout<<&angka<<" memiliki nilai "<<angka<<endl;
16
17     return 0;
18 }
```

0x7ffd5f08a4f0 memiliki nilai 50

ptr = 0x7ffd5f08a4f0

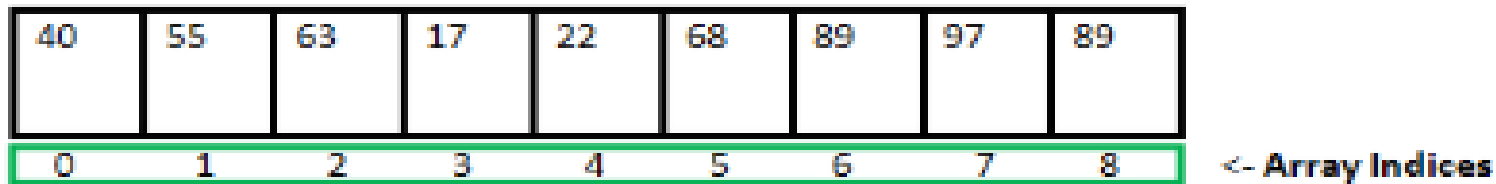
ptr = 0x7ffd5f08a4f4

0x7ffd5f08a4f4 memiliki nilai 1

...Program finished with exit code 0
Press ENTER to exit console.

Linked List (Senarai Berkait)

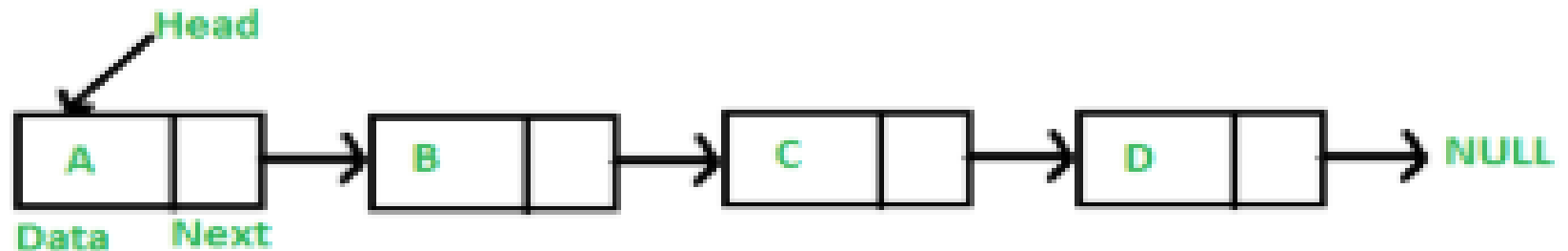
- Seperti array, Linked List adalah struktur data linier, namun elemen-elemennya tidak disimpan di lokasi yang berdekatan tetapi antar elemen dihubungkan menggunakan pointer.



Array Length = 9

First Index = 0

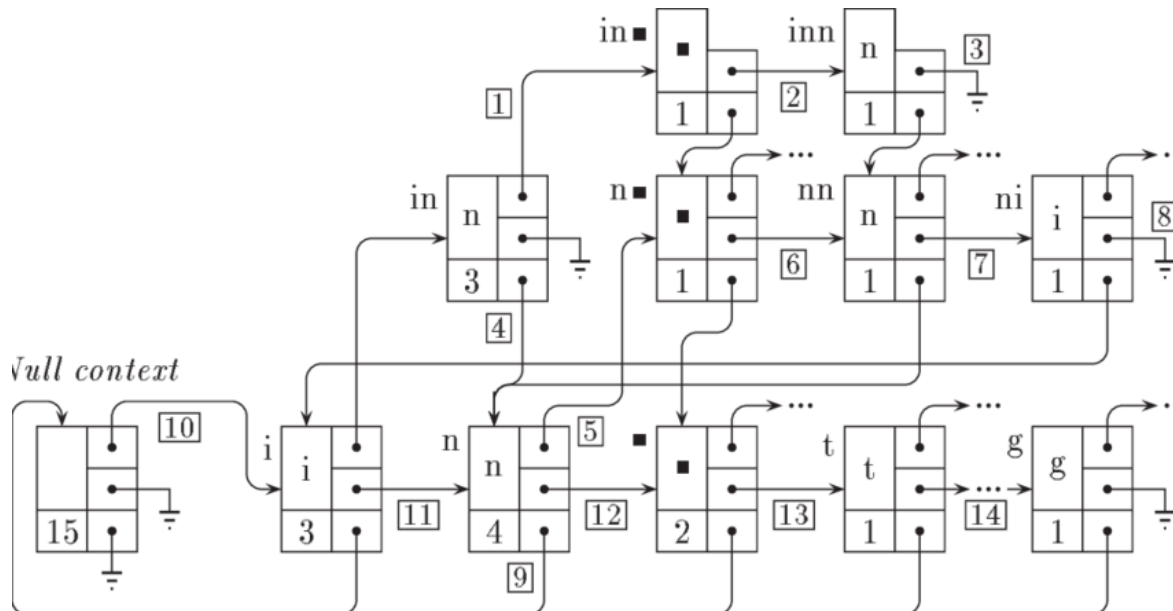
Last Index = 8



Linked List Vs Array

Kelebihan dibanding Array

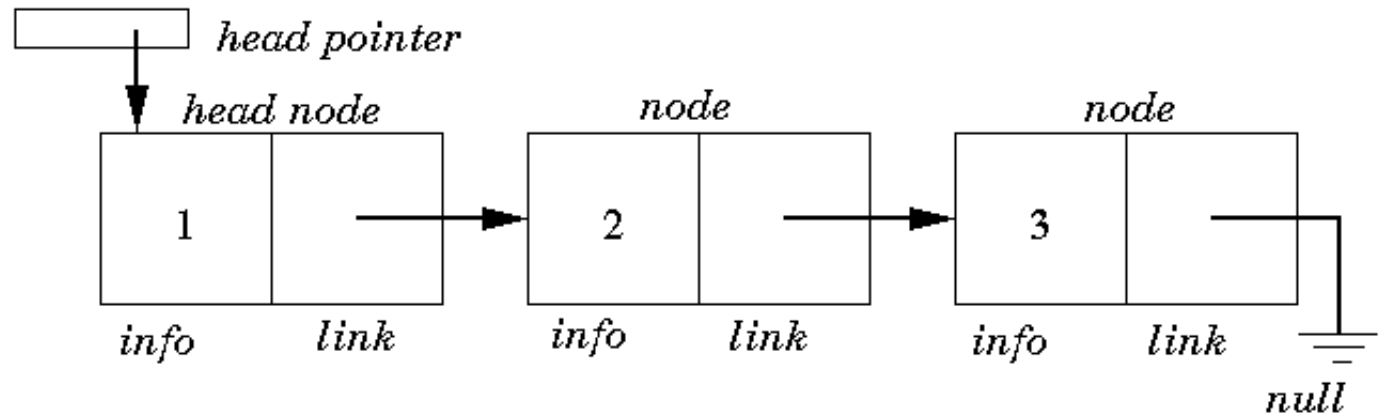
- Ukuran dinamis
- Kemudahan penyisipan / penghapusan



Kekurangan

- Akses acak tidak diizinkan. Kita harus mengakses elemen secara berurutan mulai dari node pertama.
- Ruang memori tambahan untuk sebuah pointer diperlukan pada setiap elemen dari list.
- Tidak ramah cache. Elemen-elemen dari linked list dapat ditempatkan di mana saja di memori. Jadi ketika melakukan iterasi melalui linked-list, itu akan menyebabkan banyak cache miss dan overhead kinerja.

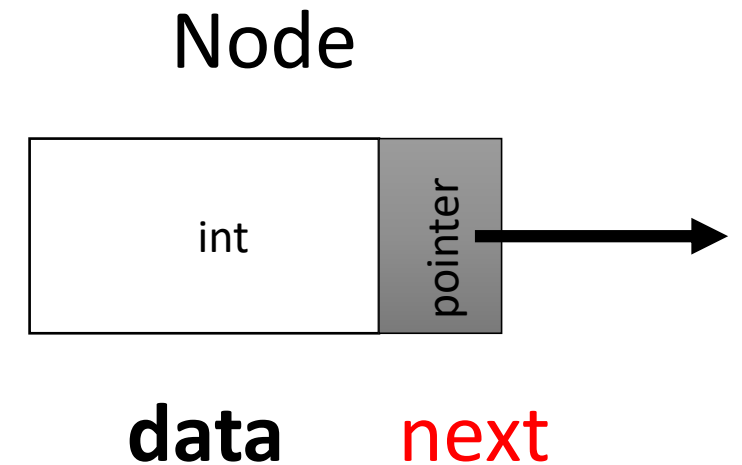
Skema Linked List



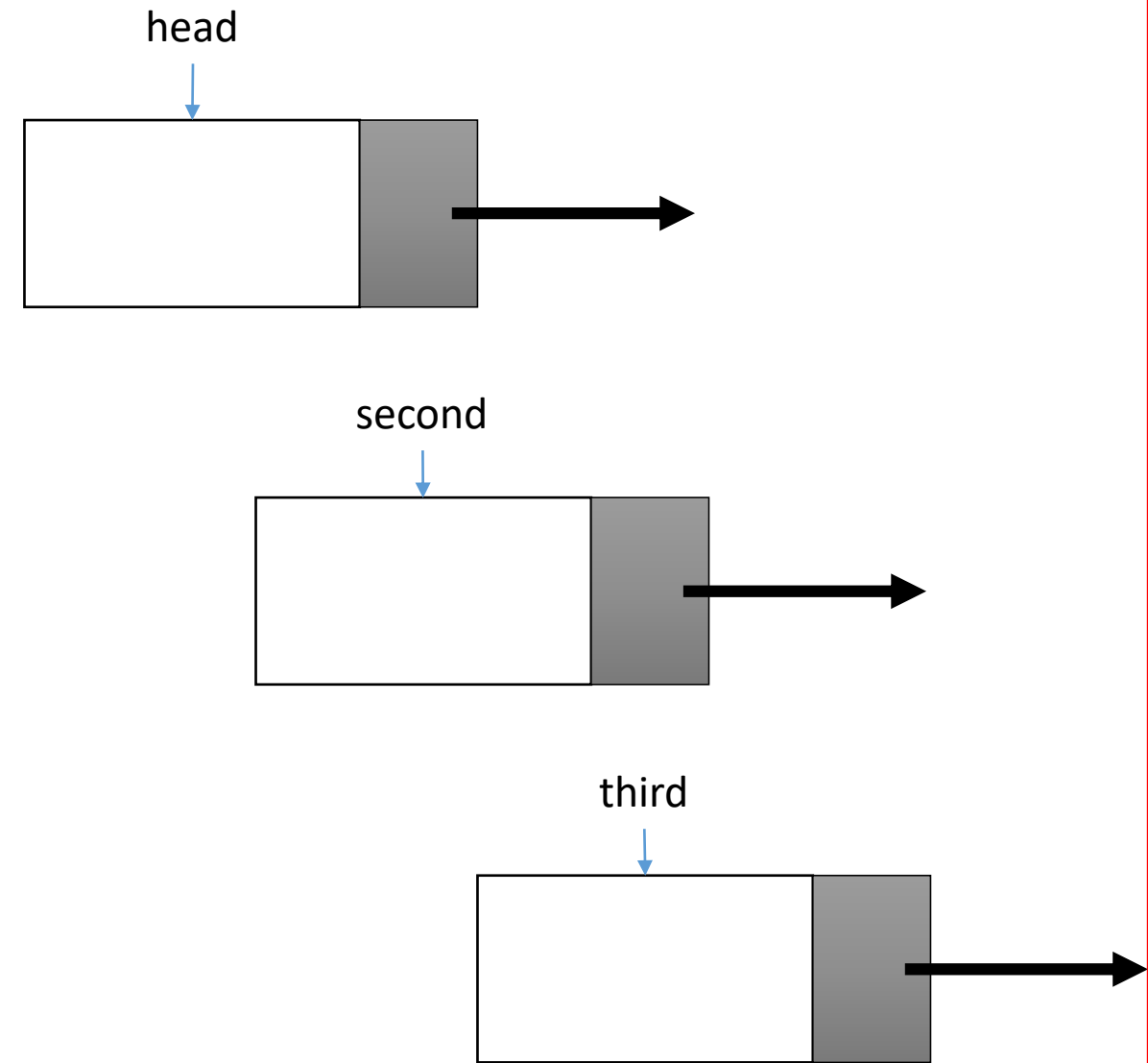
- Linked list diwujudkan dengan pointer yang menunjukan pada node pertama.
- Node pertama disebut **Head Node** (kepala)
- Node terakhir disebut **tail** (ekor/ ujung), dimana pointer-nya menunjukan **null**.
- **Node** terdiri dari:
 - **Data**
 - **Pointer** yang menunjukan pada node berikutnya
- Node bisa direpresentasikan menggunakan **struct** (bahasa C) atau kelas tersendiri (C++).

Implementasi pada C++

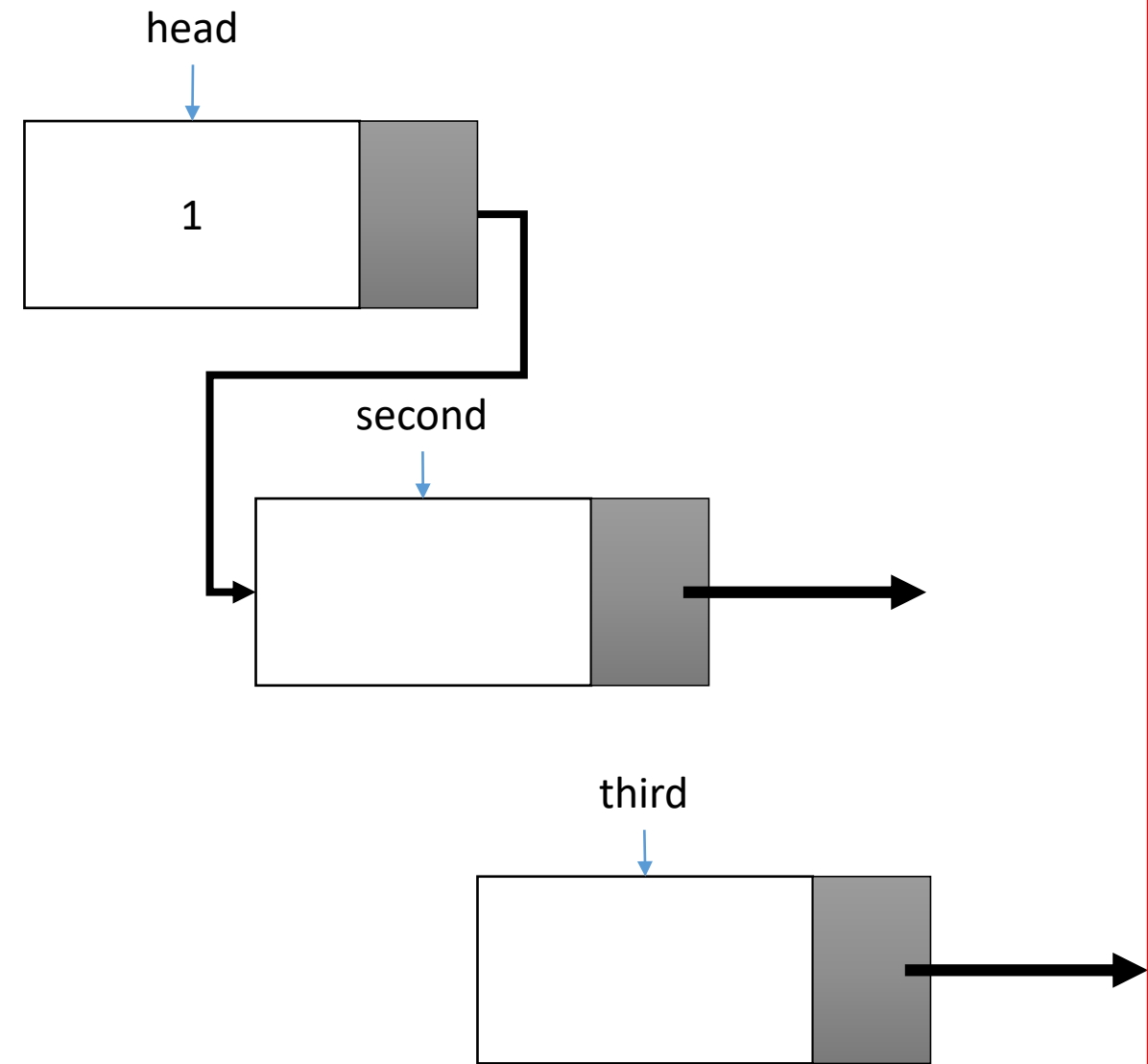
```
1  #include <iostream>
2  using namespace std;
3
4  class Node{
5      public:
6          int data;
7          Node* next;
8  };
9
```



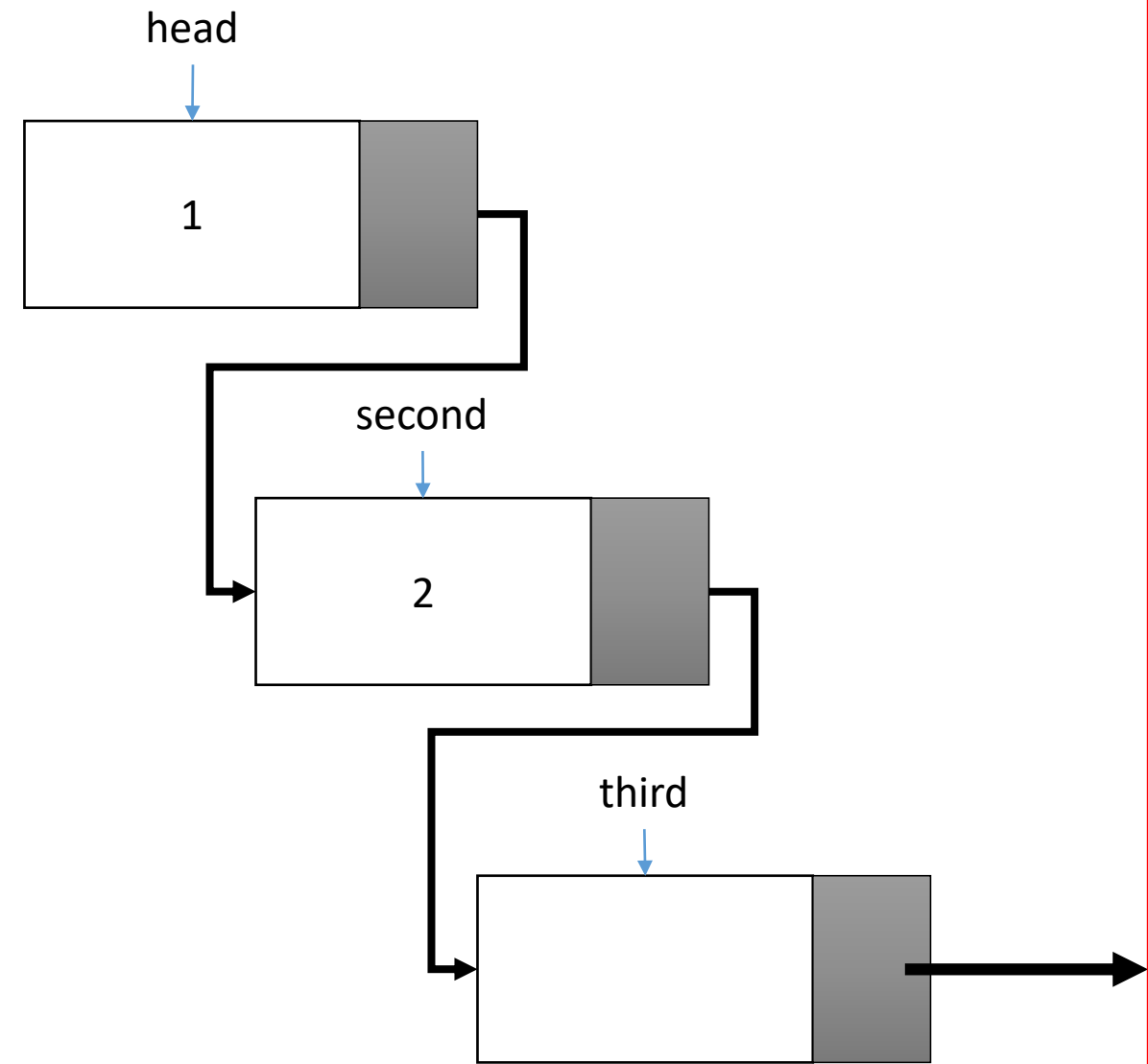
```
10 int main(){
11     Node* head = NULL;
12     Node* second = NULL;
13     Node* third = NULL;
14
15     head = new Node();
16     second = new Node();
17     third = new Node();
18
19     head->data = 1;
20     head->next = second;
21
22     second->data = 2;
23     second->next = third;
24
25     third->data = 3;
26     third->next = NULL;
27
28     return 0;
29 }
```



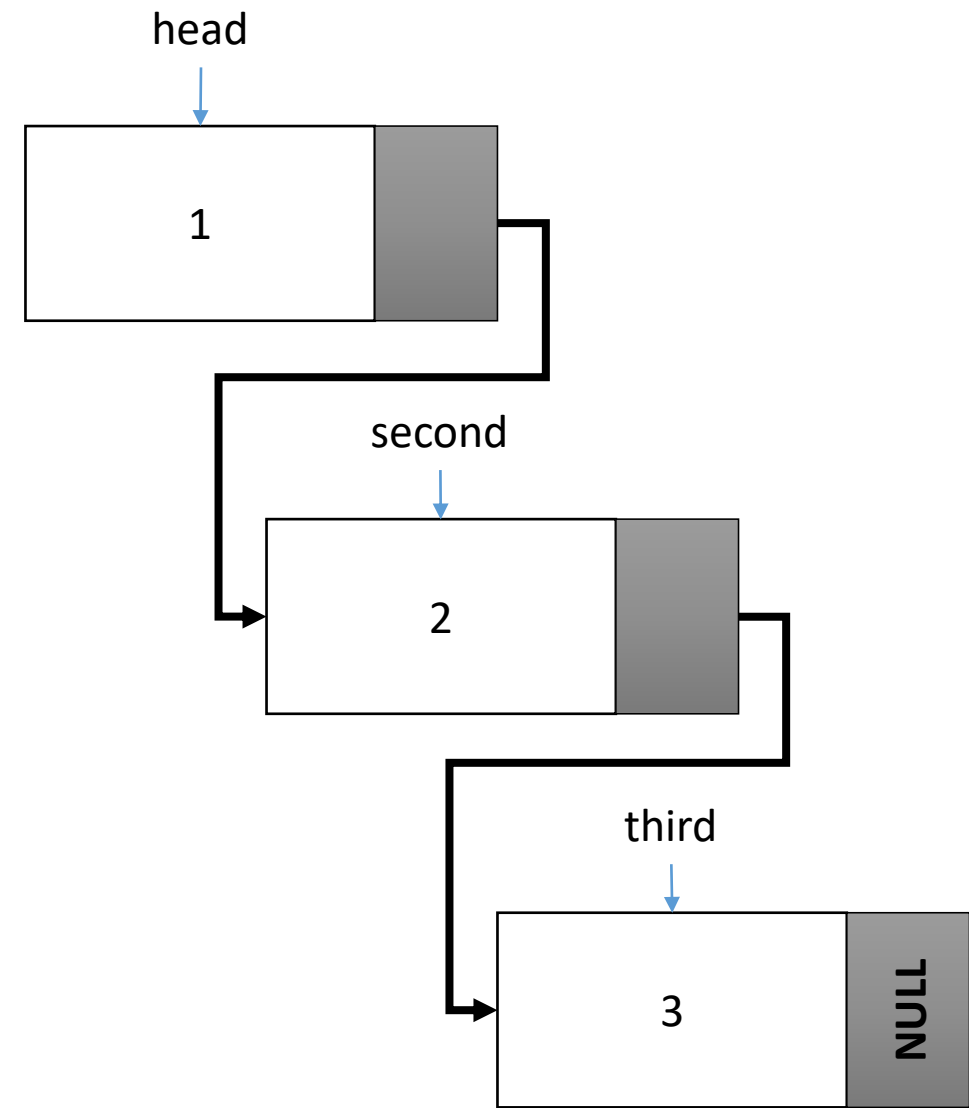
```
10 int main(){
11     Node* head = NULL;
12     Node* second = NULL;
13     Node* third = NULL;
14
15     head = new Node();
16     second = new Node();
17     third = new Node();
18
19     head->data = 1;
20     head->next = second;
21
22     second->data = 2;
23     second->next = third;
24
25     third->data = 3;
26     third->next = NULL;
27
28     return 0;
29 }
```



```
10 int main(){
11     Node* head = NULL;
12     Node* second = NULL;
13     Node* third = NULL;
14
15     head = new Node();
16     second = new Node();
17     third = new Node();
18
19     head->data = 1;
20     head->next = second;
21
22     second->data = 2;
23     second->next = third;
24
25     third->data = 3;
26     third->next = NULL;
27
28     return 0;
29 }
```



```
10 int main(){
11     Node* head = NULL;
12     Node* second = NULL;
13     Node* third = NULL;
14
15     head = new Node();
16     second = new Node();
17     third = new Node();
18
19     head->data = 1;
20     head->next = second;
21
22     second->data = 2;
23     second->next = third;
24
25     third->data = 3;
26     third->next = NULL;
27
28     return 0;
29 }
```



Melintasi Linked List

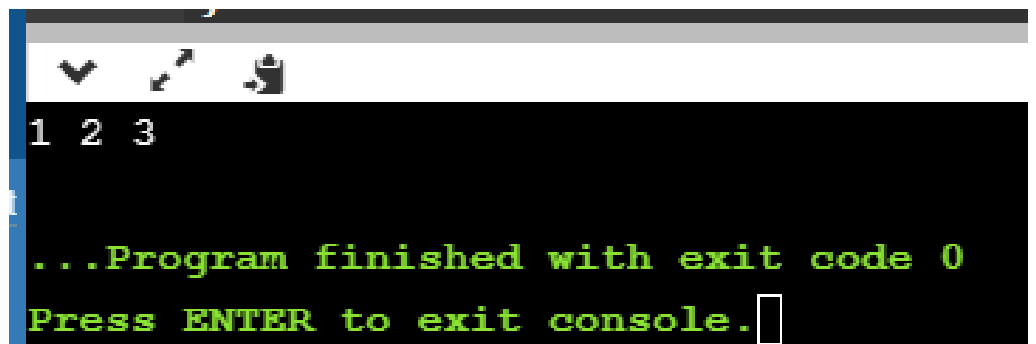
- Mari kita menelusuri daftar yang dibuat dan mencetak data dari setiap node.
- Untuk traversal, mari kita menulis printlist fungsi umum () yang mencetak daftar yang diberikan.

```
119 // This function prints contents of linked list
120 // starting from the given node
121 void printList(Node* n)
122 {
123     while (n != NULL) {
124         cout << n->data << " ";
125         n = n->next;
126     }
127     cout<<"\n\n";
128 }
129
```

Tambahkan pada code program sebelumnya. Simpan sebelum method **main()**

Panggil function dari main()

```
20 int main(){
21     Node* head = NULL;
22     Node* second = NULL;
23     Node* third = NULL;
24
25     head = new Node();
26     second = new Node();
27     third = new Node();
28
29     head->data = 1;
30     head->next = second;
31
32     second->data = 2;
33     second->next = third;
34
35     third->data = 3;
36     third->next = NULL;
37
38     printList(head);
39
40     return 0;
41 }
```



```
1 2 3
...Program finished with exit code 0
Press ENTER to exit console.
```


Insert Data pada Linked List

3 cara menambahkan data pada suatu linked list:

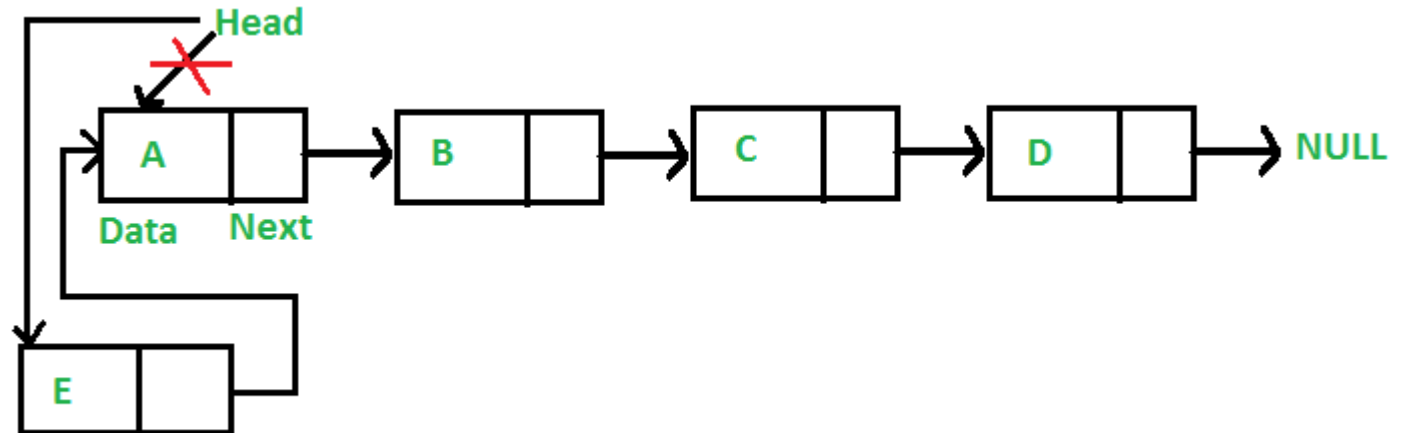
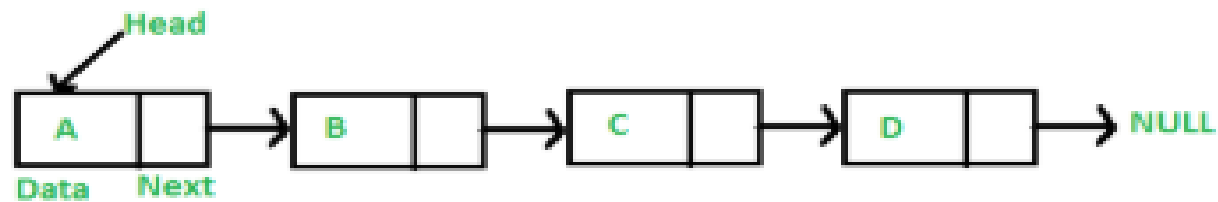
- Ditambahkan pada bagian depan

- Ditambahkan setelah node tertentu

- Ditambahkan pada bagian akhir

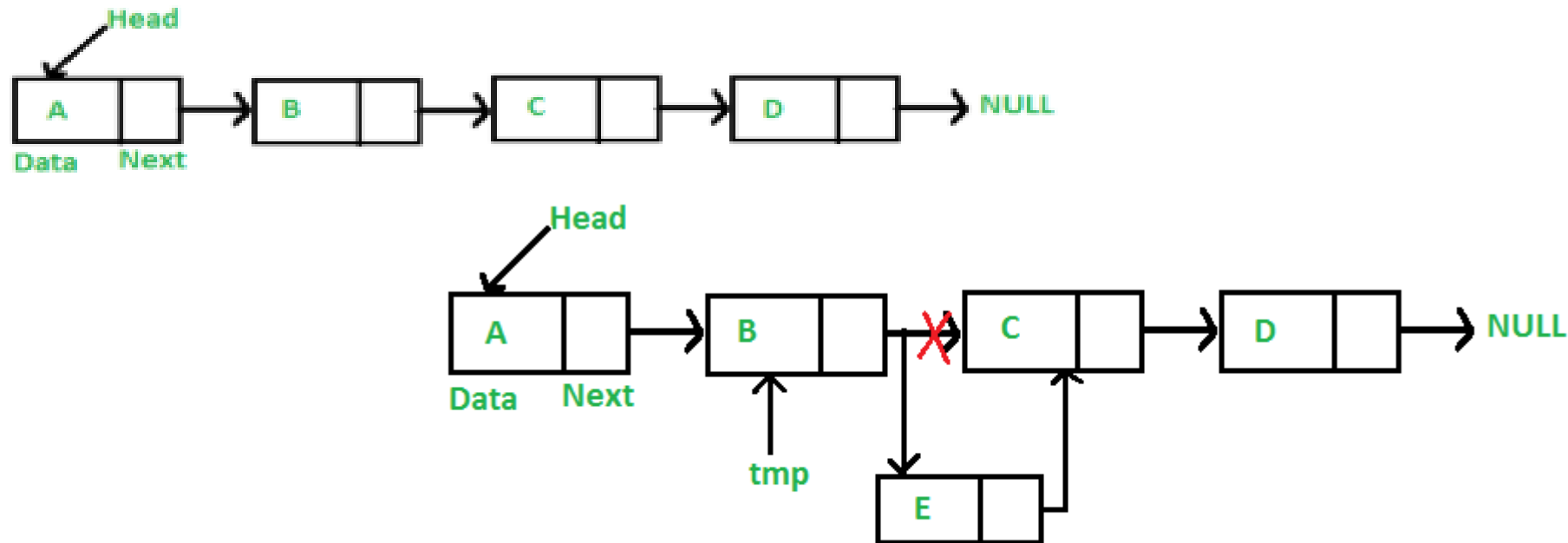
1) Menambahkan Node dibagian Depan

- Node baru selalu ditambahkan sebelum bagian **head**, kemudian node baru tersebut menjadi **head** baru dari Linked List tersebut.



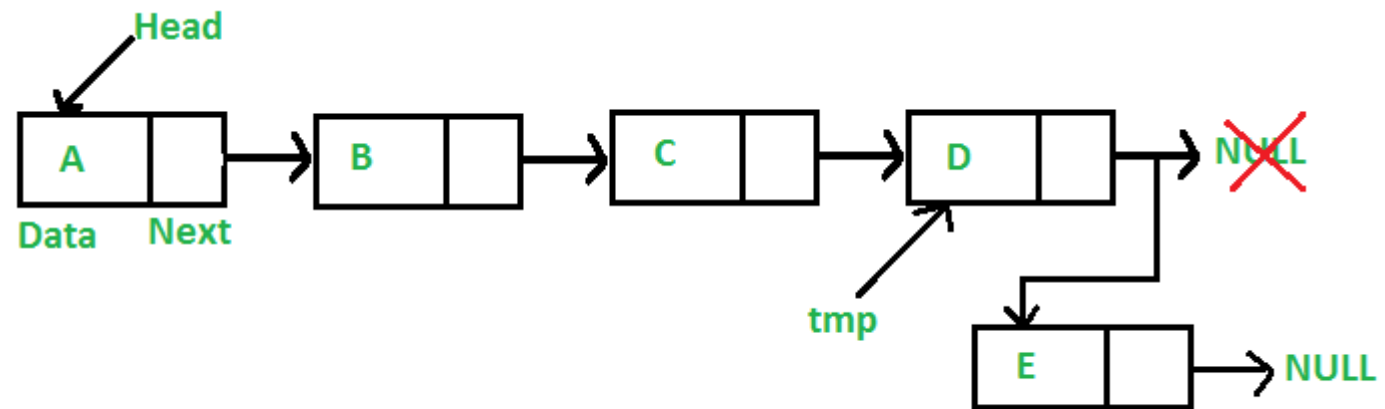
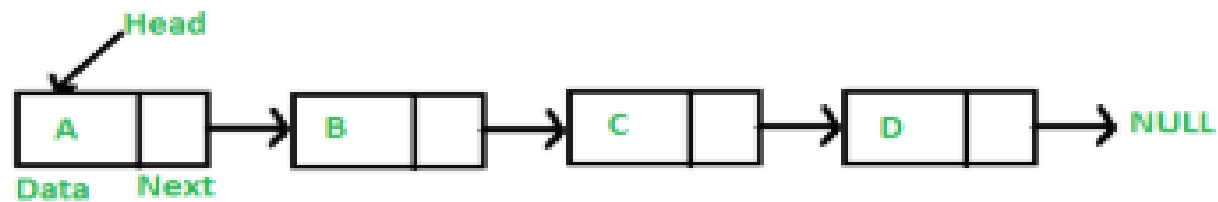
2) Menambahkan Node setelah Node tertentu

- Pointer yang mengarahkan pada node terpilih diarahkan ke sebuah node baru
- node baru dimasukkan setelah node yang diberikan.



3) Menambahkan Node dibagian Akhir

- Node baru selalu ditambahkan sebelum bagian **tail**, kemudian node baru tersebut menjadi **tail** baru dari Linked List tersebut.



Implementasi pada C++

1) Menambahkan pada bagian Depan

```
10 //function menambahkan node
11 // 1) menambahkan didepan
12
13 int tambahDepan(Node** head_ref, int data_baru){
14     //a) siapkan Node baru
15     Node* node_baru = new Node();
16
17     //b) masukan data_baru pada node_baru
18     node_baru->data = data_baru;
19
20     //c) buat node_baru jadi head
21     node_baru->next = (*head_ref);
22
23     //d) rubah head agar mengarah ke node_baru
24     (*head_ref) = node_baru;
25
26     return 0;
27 }
```

Tambahkan pada code program sebelumnya.

Simpan sebelum method **main()**

Panggil function dari main()

```
58
59     tambahDepan(&head, 30);|
60     cout<<"Setelah penambahan data didepan \n";
61     printList(head);
62                                     Tambahkan pada main() code
63     return 0; ← program sebelumnya.
64 }                                     Simpan sebelum return o;
```

1 2 3 Setelah penambahan data didepan

30 1 2 3 ← Isi Linked List setelah penambahan

...Program finished with exit code 0

Press ENTER to exit console.

2) Menambahkan setelah Node terpilih

```
28 // 2) menambahkan setelah node tertentu
29 int sisipkanSetelah(Node* node_terpilih,
30     int data_baru){
31     //a) cek apakah node_terpilih adalah NULL
32     if(node_terpilih == NULL){
33         cout<<"Node yang dipilih KOSONG. Penambahan Batal\n";
34         return 0;
35     }
36
37     //b) menyiapkan Node Baru
38     Node* node_baru = new Node();
39
40     //c) masukan data_baru pada node_baru
41     node_baru->data = data_baru;
42
43     //d) buat node_baru mengarahkan pada node
44     // yang sama diarahkan node_terpilih
45     node_baru->next = node_terpilih->next;
46
47     //e) ubah node_terpilih mengarahkan ke node_baru
48     node_terpilih->next = node_baru;
49
50     return 0;
51 }
```

Tambahkan pada code program sebelumnya.

Simpan sebelum method **main()**

Panggil function dari main()

```
84     tambahDepan(&head, 30);
85     cout<<"Setelah penambahan data didepan \n";
86     printList(head);
87
88     sisipkanSetelah(head->next, 50);
89     cout<<"Setelah penambahan data setelah First \n";
90     printList(head);
91
92     return 0;
93 }
```

Tambahkan pada **main()** code program sebelumnya.
Simpan sebelum return o;

1 2 3 Setelah penambahan data didepan
30 1 2 3 Setelah penambahan data setelah First
30 1 50 2 3

Isi Linked List setelah penambahan

3) Menambahkan pada bagian Akhir

```
52
53 // 3) menambahkan pada bagian akhir
54 int tambahAkhir(Node** head_ref, int data_baru){
55     //a) siapkan node_baru
56     Node* node_baru = new Node();
57     Node *last = *head_ref;
58
59     //b) masukan data_baru pada node_baru
60     node_baru->data = data_baru;
61
62     //c) menjadikan node_baru sbg last node
63     node_baru->next = NULL;
64
65     //d) mengecek apakah list kosong?
66     if(*head_ref == NULL){
67         //jika kosong, node_baru jd head
68         *head_ref = node_baru;
69
70         return 0;
71     }
```

Tambahkan pada code program sebelumnya.

Simpan sebelum method **main()**

3) Menambahkan pada bagian Akhir (2)

```
72
73     //e) jika tidak kosong, jelajahi list sampai last
74     while(last->next != NULL){
75         last = last->next;
76     }
77
78     //f) rubah Last node mengarahkan ke node_baru
79     last->next = node_baru;
80
81     return 0;
82 }
```

Tambahkan pada code program sebelumnya.

Simpan sebelum method **main()**

Panggil function dari main()

```
115 tambahDepan(&head, 30);
116 cout<<"Setelah penambahan data didepan \n";
117 printList(head);
118
119 sisipkanSetelah(head->next, 50);
120 cout<<"Setelah penambahan data setelah First \n";
121 printList(head);
122
123 tambahAkhir(&head, 70);
124 cout<<"Setelah penambahan data dibelakang \n";
125 printList(head);
126
127 return 0;
128 }
129
```

Tambahkan pada **main()**
code program sebelumnya.
Simpan sebelum return 0;



input

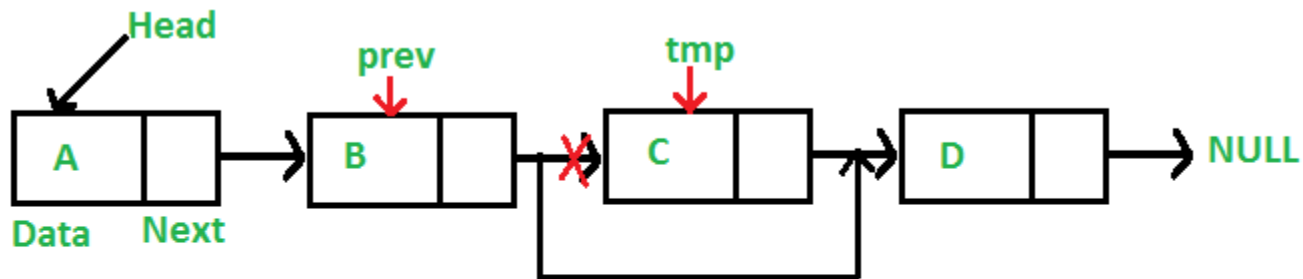
```
2 3 Setelah penambahan data didepan
0 1 2 3 Setelah penambahan data setelah First
0 1 50 2 3 Setelah penambahan data dibelakang
0 1 50 2 3 70
```

Isi Linked List setelah
penambahan

Delete Node pada Linked List

Proses Delete berdasarkan *Key*

- Diberikan '*kunci*', hapus kemunculan pertama dari kunci ini di Linked List.
- Untuk menghapus Node dari Linked List, perlu dilakukan langkah-langkah berikut:
 1. Temukan Node sebelumnya dari Node yang akan dihapus.
 2. Ubah next Node sebelumnya sama seperti next Node yang akan dihapus.
 3. Kosongkan memori untuk Node yang akan dihapus.



Implementasi pada C++

```
84 //parameter input reference Head & key
85 int hapusNodeKey(Node **head_ref, int key){
86     //simpan reference head node & node sebelum
87     Node* temp = *head_ref, *node_sebelum;
88
89     //jika key berada di head node,
90     //maka headnya dihapus
91     if(temp != NULL && temp->data == key){
92         *head_ref = temp->next; //merubah head
93         free(temp); //membebaskan head Lama
94
95         return 0;
96     }
```

Tambahkan pada code program sebelumnya.
Simpan sebelum method **main()**

```
98 //mencari key node yang akan didelet,  
99 //terus mendeteksi node sebelumnya  
100 while(temp != NULL && temp->data != key){  
101     node_sebelum = temp;  
102     temp = temp->next;  
103 }  
104  
105 //jika key tidak ditemukan di linked list  
106 //proses berhenti  
107 if(temp == NULL) return 0;  
108  
109 //melepaskan node key dari linked list  
110 node_sebelum->next = temp->next;  
111  
112 //free memory  
113 free(temp);  
114  
115 return 0;  
116 }
```


Panggil function `hapusNodeKey()` di `main()`

```
150     tambahDepan(&head, 30);
151     cout<<"Setelah penambahan data didepan \n";
152     printList(head);
153
154     sisipkanSetelah(head->next, 50);
155     cout<<"Setelah penambahan data setelah First \n";
156     printList(head);
157
158     tambahAkhir(&head, 70);
159     cout<<"Setelah penambahan data dibelakang \n";
160     printList(head);
161
162     int key = 30;
163     hapusNodeKey(&head, key);
164     cout<<"Setelah "<<key<<" dihapus \n";
165     printList(head);
166
167     return 0;
168 }
```

Tambahkan pada `main()`
code program sebelumnya.
Simpan sebelum `return 0;`

Running Program

1 2 3

Setelah penambahan data didepan

30 1 2 3

Setelah penambahan data setelah First

30 1 50 2 3

Setelah penambahan data dibelakang

30 1 50 2 3 70

Setelah 30 dihapus

1 50 2 3 70

...Program finished with exit code 0

Press ENTER to exit console.

Delete Node berdasarkan **Posisi**

- Jika simpul yang akan dihapus adalah **head**, cukup lakukan proses hapus saja.
- Untuk menghapus node yang berada ditengah list, kita harus memiliki pointer ke Node sebelum Node yang akan dihapus.
- Jadi jika posisi tidak nol, kita menjalankan posisi loop-1 kali dan mendapatkan pointer ke node sebelumnya.

Fungsi hapusNodePosisi()

```
118 //parameter input reference head & posisi
119 //hapus node pada posisi tersebut
120 int hapusNodePosisi(Node **head_ref, int posisi){
121     //jika list kosong
122     if(*head_ref == NULL) return 0; //proses selesai
123
124     //simpan node head
125     Node* temp = *head_ref;
126
127     //jika posisi adalah head
128     if(posisi == 0){
129         *head_ref = temp->next; //rubah head
130         free(temp); //melepaskan temp
131
132         return 0; //proses selesai
133     }
```

Tambahkan pada code program sebelumnya.
Simpan sebelum method main()

```
135 //mencari node sebelum node yg akan di hapus
136 for(int i=0; temp != NULL && i < posisi-1; i++){
137     temp = temp->next;
138 }
139
140 //jika posisi lebih dari jumlah node, selesai
141 if(temp == NULL || temp->next == NULL) return 0;
142
143 //node yg akan dihapus : temp->next
144 //simpan pointer pada next dari node yg akan dihapus
145 Node *node_selanjutnya = temp->next->next;
146
147 //lepaskan node dari list
148 free(temp->next); //membebaskan memory
149 temp->next = node_selanjutnya;
150
151 return 0;
152 }
```

Panggil function dari main()

```
186 tambahDepan(&head, 30);
187 cout<<"Setelah penambahan data didepan \n";
188 printList(head);
189
190 sisipkanSetelah(head->next, 50);
191 cout<<"Setelah penambahan data setelah First \n";
192 printList(head);
193
194 tambahAkhir(&head, 70);
195 cout<<"Setelah penambahan data dibelakang \n";
196 printList(head);
197
198 int key = 50;
199 hapusNodeKey(&head, key);
200 cout<<"Setelah " <<key<<" dihapus \n";
201 printList(head);
202
203 int posisi = 2;
204 hapusNodePosisi(&head, posisi);
205 cout<<"Setelah Node ke-" <<posisi<<" dihapus \n";
206 printList(head);
207
208 return 0;
209 }
```

Tambahkan pada **main()**
code program sebelumnya.
Simpan sebelum return o;

Running Program

```
1 2 3
```

```
Setelah penambahan data didepan
```

```
30 1 2 3
```

```
Setelah penambahan data setelah First
```

```
30 1 50 2 3
```

```
Setelah penambahan data dibelakang
```

```
30 1 50 2 3 70
```

```
Setelah 50 dihapus
```

```
30 1 2 3 70
```

```
Setelah Node ke-2 dihapus
```

```
30 1 3 70
```

```
...Program finished with exit code 0
```

```
Press ENTER to exit console.
```

Delete Linked List

Menghapus Sebuah Linked List

- Algoritma Untuk C / C ++:
 - Iterasi melalui linked list dan hapus semua node satu per satu.
 - Poin utama di sini, tidak mengakses pointer saat ini jika pointer saat ini dihapus.
- Di Java, terdapat *automatic garbage collection*, jadi menghapus linked list itu mudah, hanya perlu mengubah **head** menjadi **null**.

Implementasi pada C++

```
154 //menghapus seluruh linked list
155 int hapusList(Node **head_ref){
156     //ubah reference head untuk head_ref
157     Node* node_saatIni = *head_ref;
158     Node* node_selanjutnya;
159
160     //menghapus node satu per satu
161     while(node_saatIni != NULL){
162         node_selanjutnya = node_saatIni->next;
163         free(node_saatIni);
164         node_saatIni = node_selanjutnya;
165     }
166
167     //mengubah reference head
168     *head_ref = NULL;
169
170     return 0;
171 }
```

Tambahkan pada code program sebelumnya.
Simpan sebelum method **main()**

Panggil function `hapusList()` pada `main()`

```
216
217     int key = 50;
218     hapusNodeKey(&head, key);
219     cout<<"Setelah "<<key<<" dihapus \n";
220     printList(head);
```

```
221
222     int posisi = 2;
223     hapusNodePosisi(&head, posisi);
224     cout<<"Setelah Node ke-"<<posisi<<" dihapus \n";
225     printList(head);
```

```
226
227     hapusList(&head);
228     cout<<"List dihapus \n";
229     printList(head);
```

```
230
231     return 0;
```

```
232 }
```

Tambahkan pada `main()`
code program sebelumnya.
Simpan sebelum `return 0`;

Running Program

```
1 2 3
```

```
Setelah penambahan data didepan
```

```
30 1 2 3
```

```
Setelah penambahan data setelah First
```

```
30 1 50 2 3
```

```
Setelah penambahan data dibelakang
```

```
30 1 50 2 3 70
```

```
Setelah 50 dihapus
```

```
30 1 2 3 70
```

```
Setelah Node ke-2 dihapus
```

```
30 1 3 70
```

```
List dihapus
```

```
...Program finished with exit code 0
```

```
Press ENTER to exit console. 
```

Menghitung Ukuran Linked List

Ukuran/ Panjang dari Linked List

- 1) Inisialisasi **jumlah** sebagai 0
- 2) Inisialisasi pointer node, node saat ini = head node
- 3) Lakukan selama node saat ini bukan NULL
 - a) node saat ini = node yang di-reference node saat ini
 - b) **jumlah** bertambah 1;
- 4) tampilkan **jumlah**


Implementasi pada C++

```
173 //menghitung panjang linked list
174 int getLength(Node* head){
175     int count = 0;
176     Node* node_saatIni = head;
177
178     while(node_saatIni != NULL){
179         count++;
180         node_saatIni = node_saatIni->next;
181     }
182
183     return count;
184 }
```

Tambahkan pada code program sebelumnya.
Simpan sebelum method **main()**

Ubah function **printList()** sebelumnya

```
187 // This function prints contents of linked list
188 // starting from the given node
189 void printList(Node* n) {
190     cout<<getLength(n)<<" node = ";
191     while (n != NULL) {
192         cout << n->data << " ";
193         n = n->next;
194     }
195     cout<<"\n\n";
196 }
```



panggil **getLength()**

Running Program

```
3 node = 1 2 3
```

```
Setelah penambahan data didepan
```

```
4 node = 30 1 2 3
```

```
Setelah penambahan data setelah First
```

```
5 node = 30 1 50 2 3
```

```
Setelah penambahan data dibelakang
```

```
6 node = 30 1 50 2 3 70
```

```
Setelah 50 dihapus
```

```
5 node = 30 1 2 3 70
```

```
Setelah Node ke-2 dihapus
```

```
4 node = 30 1 3 70
```

```
List dihapus
```

```
0 node =
```

```
...Program finished with exit code 0
```

```
Press ENTER to exit console.
```

Dikatakan tadi, bahwa akses pada linked list harus berurutan (tidak bisa acak).
Bagaimanakah proses **pencarian elemen** tercepat pada linked list?

Studi kasus