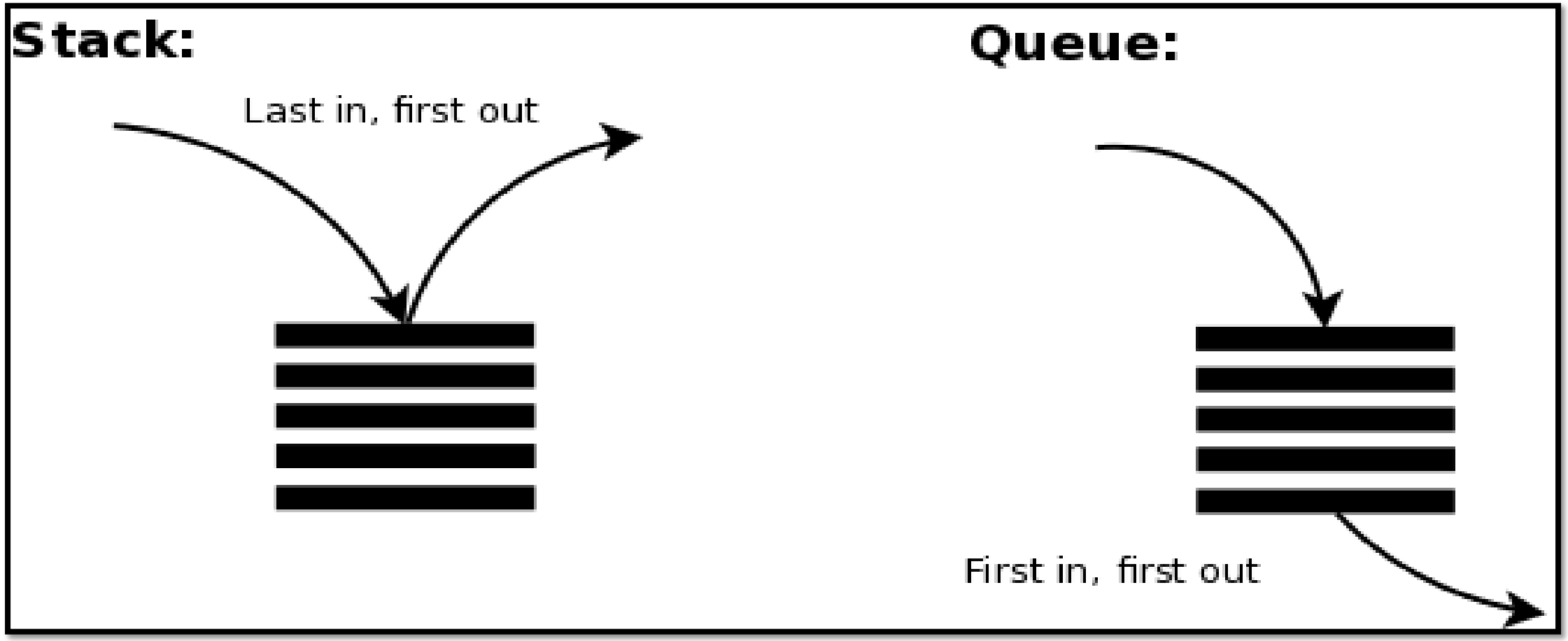


Stack & Queue Menggunakan Linked List

Fitri Nuraeni, M.Kom | Struktur Data | Reguler Pagi | 2020

Nufi3 - Last Meeting Event

Learning with nufi3.com

NUFI3 research

Data Structure

BACK TO THE TOPIC

Kembali Ke Materi

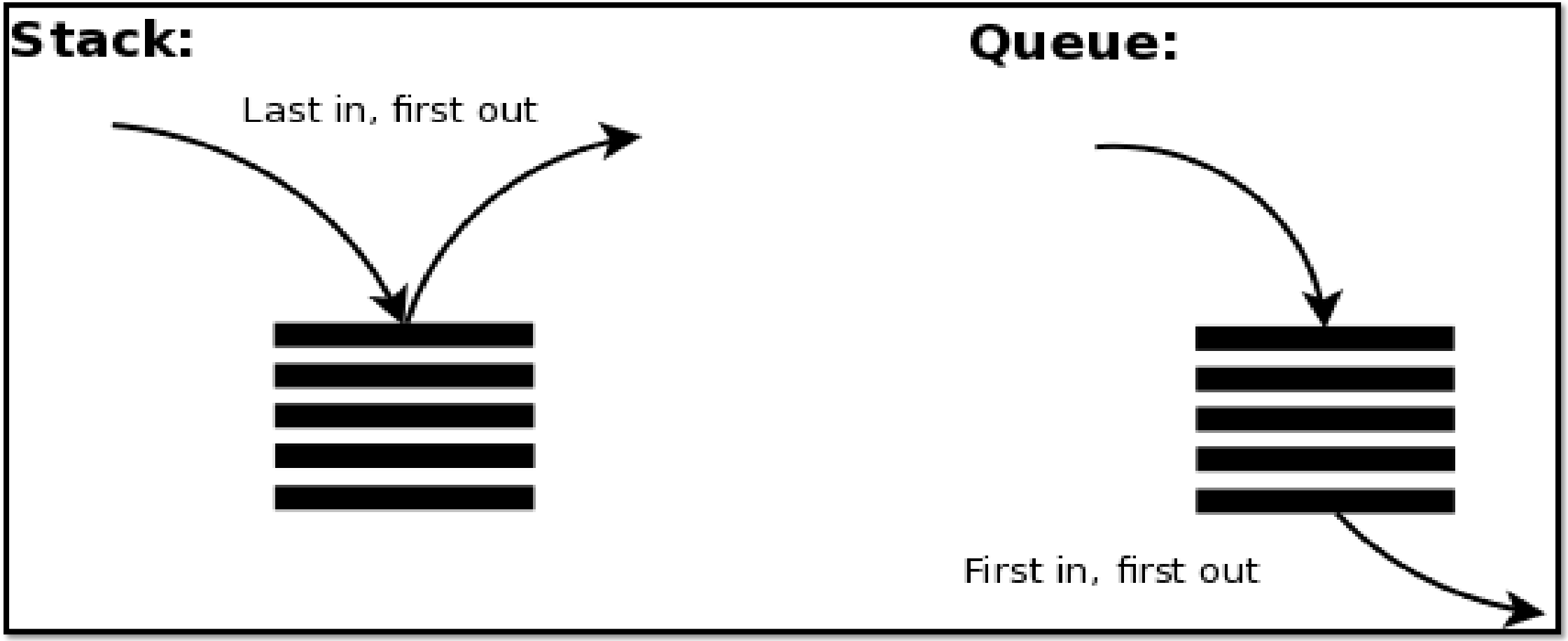
Rewards untuk event akan diumumkan di akhir pertemuan

Pertanyaan :

-Buka browser www.menti.com

- masukan kode 70 11 31

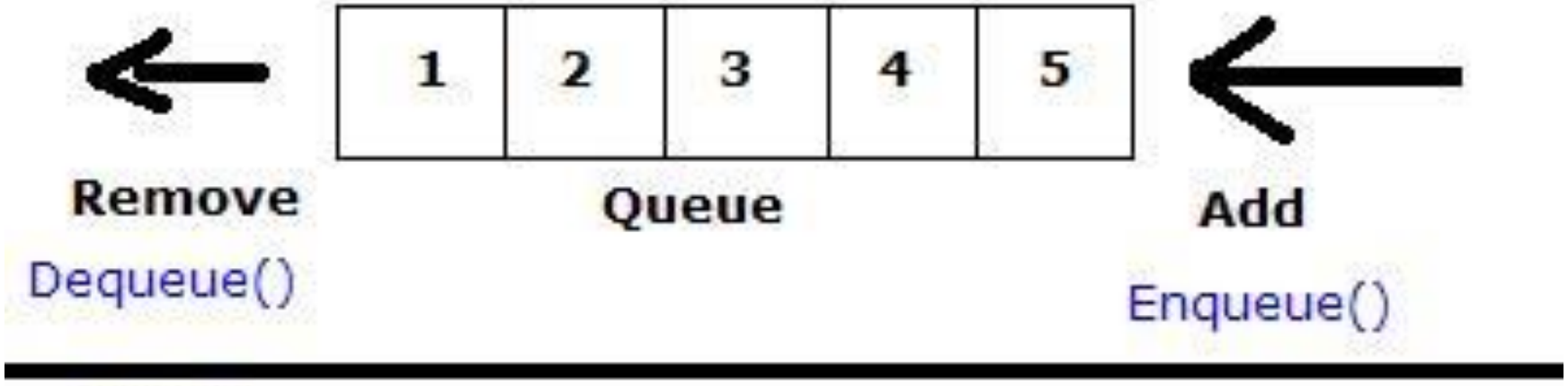
Go to www.menti.com and use the code 70 11 31



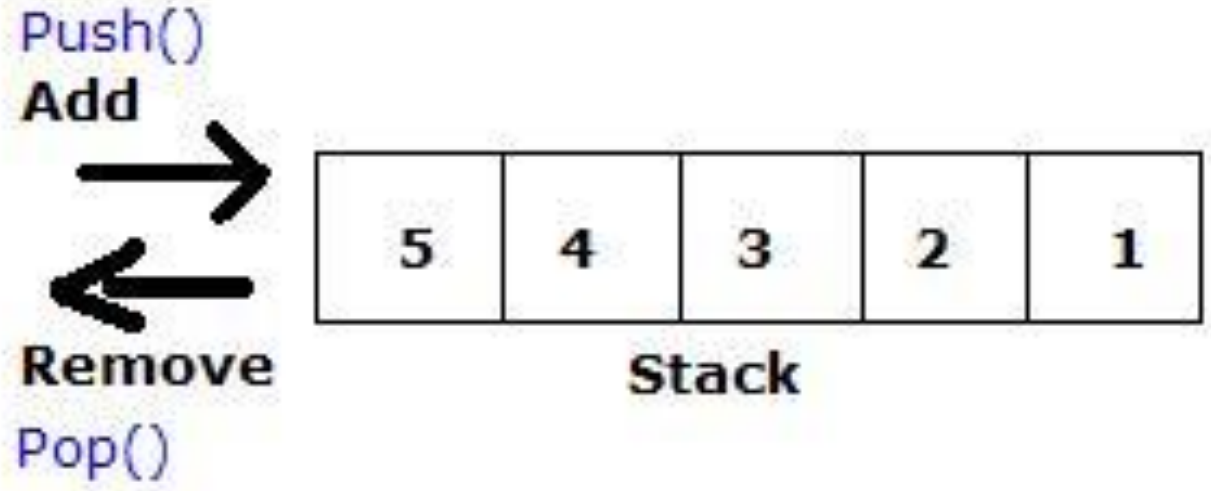
Stack & Queue Menggunakan Linked List

Mengingat kembali

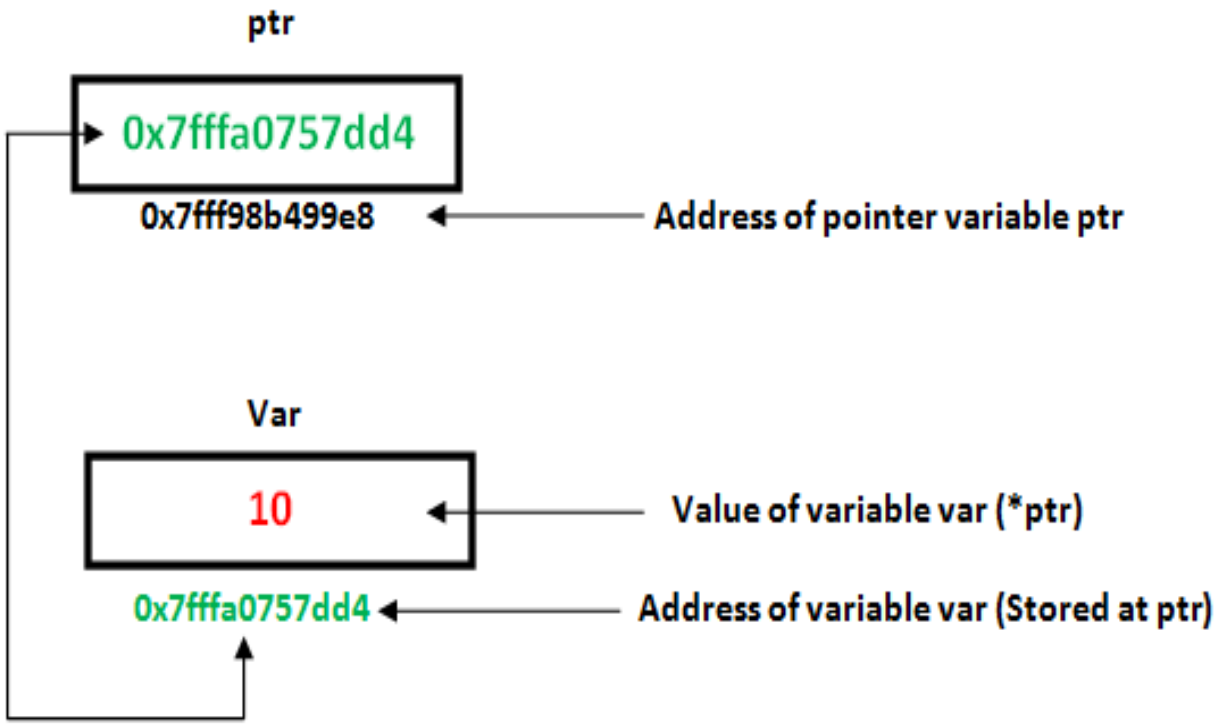
FIFO



LIFO



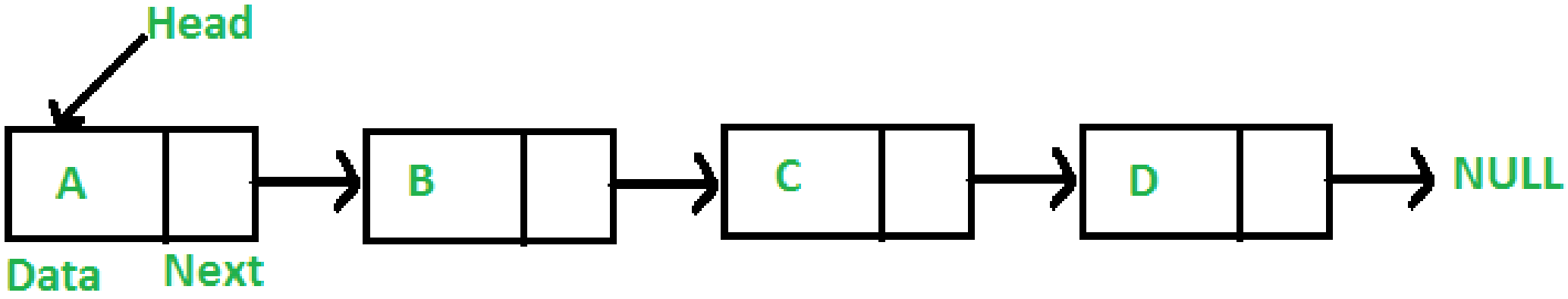
Pointer & Variabel Pointer




```
13 int main(){
14     int varAngka = 50;
15     int *pointerAngka = &varAngka;
16
17     cout << "varAngka berisi " << varAngka;
18     cout << " beralamat pada " << pointerAngka << endl;
19
20     return 0;
21 }
```

```
varAngka berisi 50 beralamat pada 0x7ffeed028eb4
```

Linked List




```
class Node{
    public:
        string data;
        Node* next;
};
```

Terdiri dari kumpulan elemen/ NODE

```
void tampilkan(Node* awal){
    int i = 0;
    cout<<"Daftar Kota : ";
    while(awal != NULL){
        cout << awal->data << endl;
        i++;

        awal = awal->next;
    }
    cout<<" Total : "<<i<<endl;
}
```

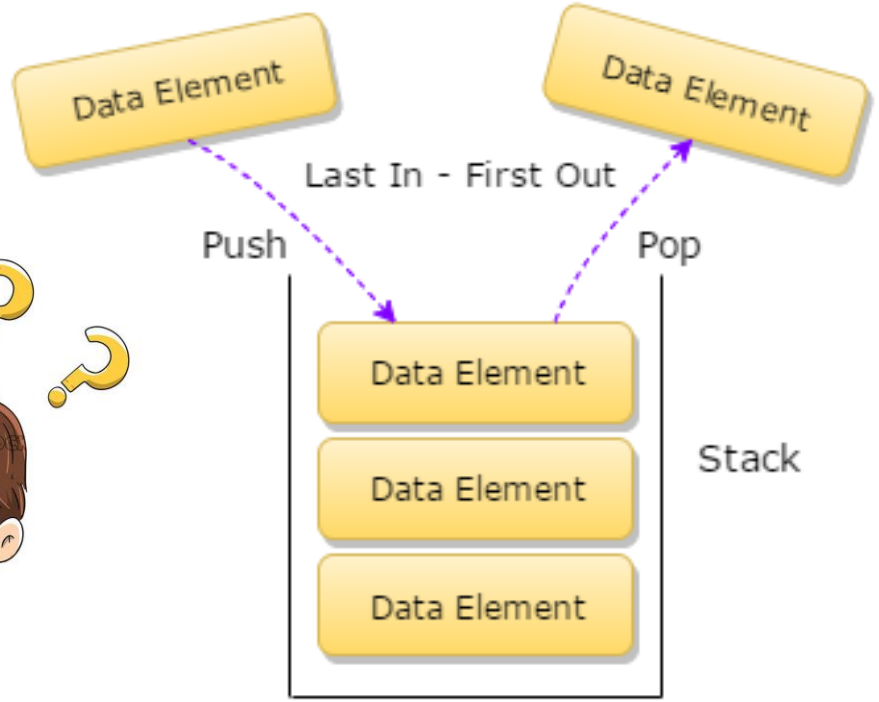
Diakses mulai dari node awal sampai akhir

```
int main(){
    Node* head = NULL;

    head = new Node();
    head->data = "tasikmalaya";
    head->next = NULL;
    tampilkan(head);
}
```

Operasi Pada Linked List

- **Tambah** Node Baru
  - Bagian depan/ Head
  - Disisipkan setelah Node tertentu
  - Bagian akhir/ Tail (last)
- **Hapus** Node
  - Berdasarkan kunci tertentur : data pada node
  - Berdasarkan posisi tertentu : dibaca mulai head
- Delete List : menghapus setiap Node sehingga Head = NULL
- Ukuran List : menghitung jumlah node dari head ke last

Bagaimana jika **stack** diimplementasikan pada Linked List?



Operasi pada Stack

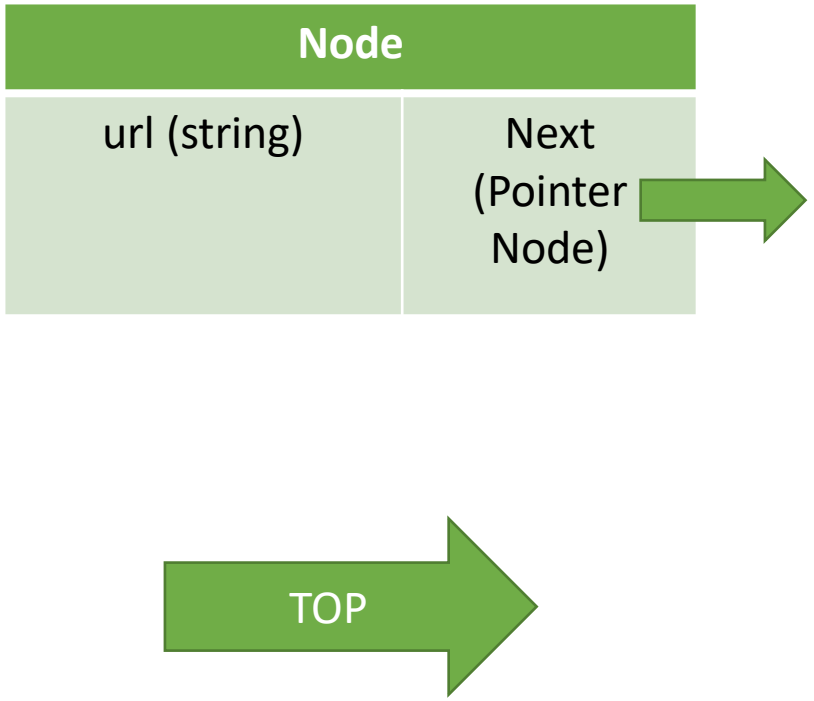
- Buat kosong : top = -1
- Cek apakah kosong : top == -1? True or false
- Cek apakah penuh : top == (N-1)? True or false
- Informasi Posisi Top : return top (indeks)
- Cetak isi stack : membaca dari Top
- Push : menambah elemen pada bagian atas/ akhir, Top++
- Pop : menghapus elemen pada bagian atas/ akhir, Top--

Operasi pada Stack menggunakan Linked List

- Buat kosong : top = -1 → top = NULL
- Cek apakah kosong : top == -1? → top == NULL : True or false
- Cek apakah penuh : top == (N-1)? True or false → **TIDAK ADA**
- Informasi Posisi Top : return top (indeks) → isi Node **Head**
- Cetak isi stack : membaca dari Top → membaca mulai Node **Head**
- Push : menambah elemen pada bagian atas/ akhir, Top++
  - Menambahkan node pada bagian depan/ HEAD
- Pop : menghapus elemen pada bagian atas/ akhir, Top—
  - Menghapus node pada bagian depan/ Head/ Posisi 0

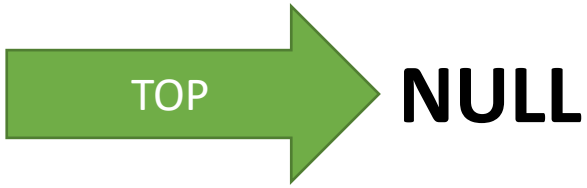
Implementasi Pada C++

```cpp
#include <iostream>

using namespace std;

class Node{
    public:
    string url;
    Node* next;
};

Node* top;

```



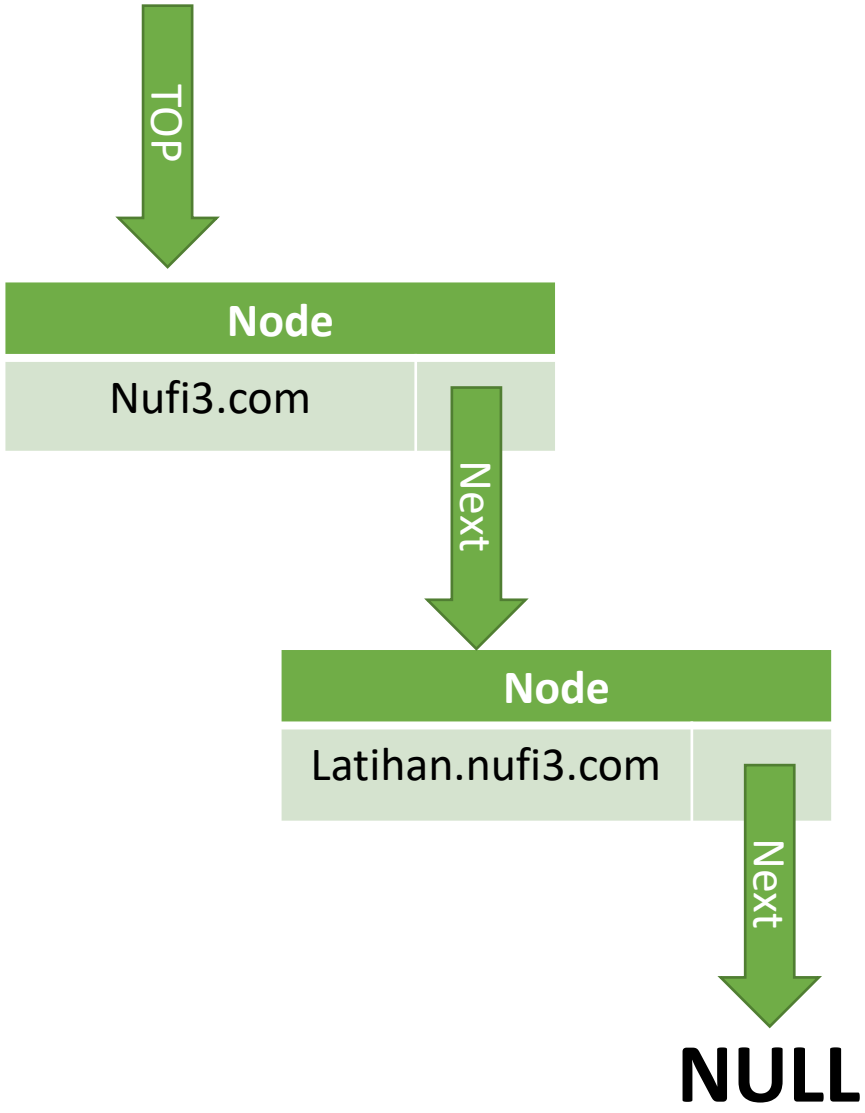
Operasi Pendukung Stack

```
13 void buatKosong(){
14     top = NULL;
15 }
16
17 bool isKosong(){
18     return (top == NULL);
19 }
20
```



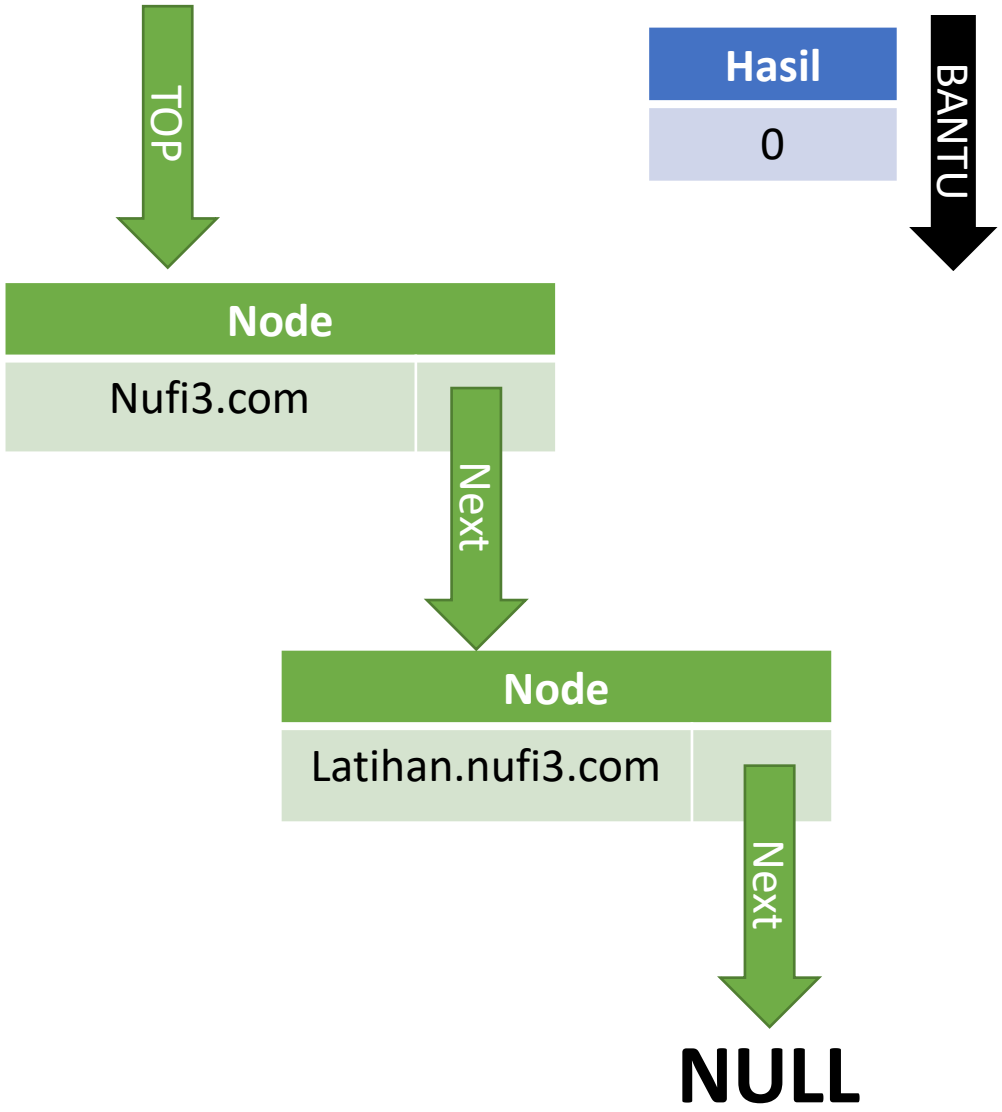
Tidak mengarah pada objek Node

KOSONG



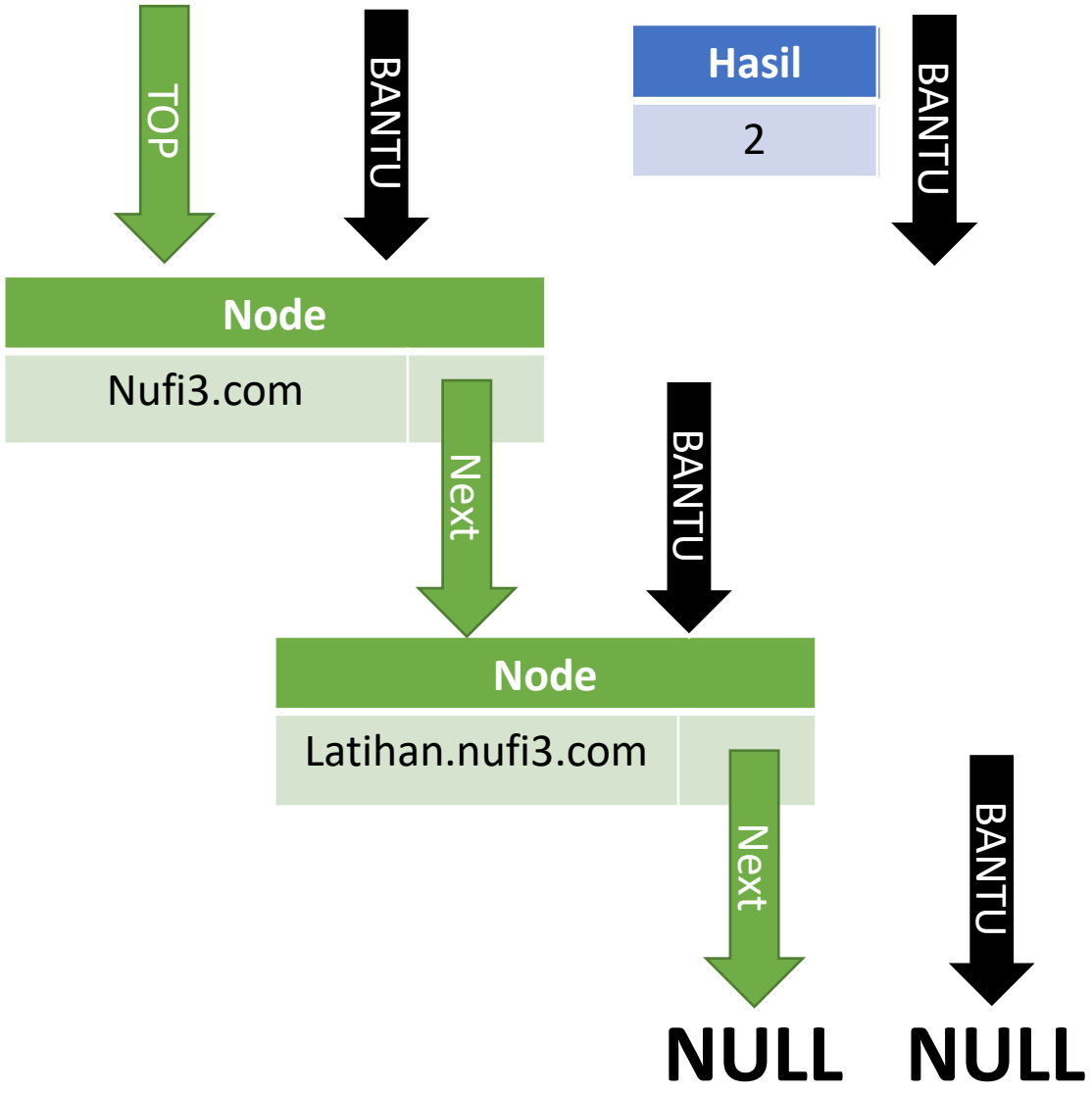
TIDAK KOSONG

Operasi Pendukung Stack



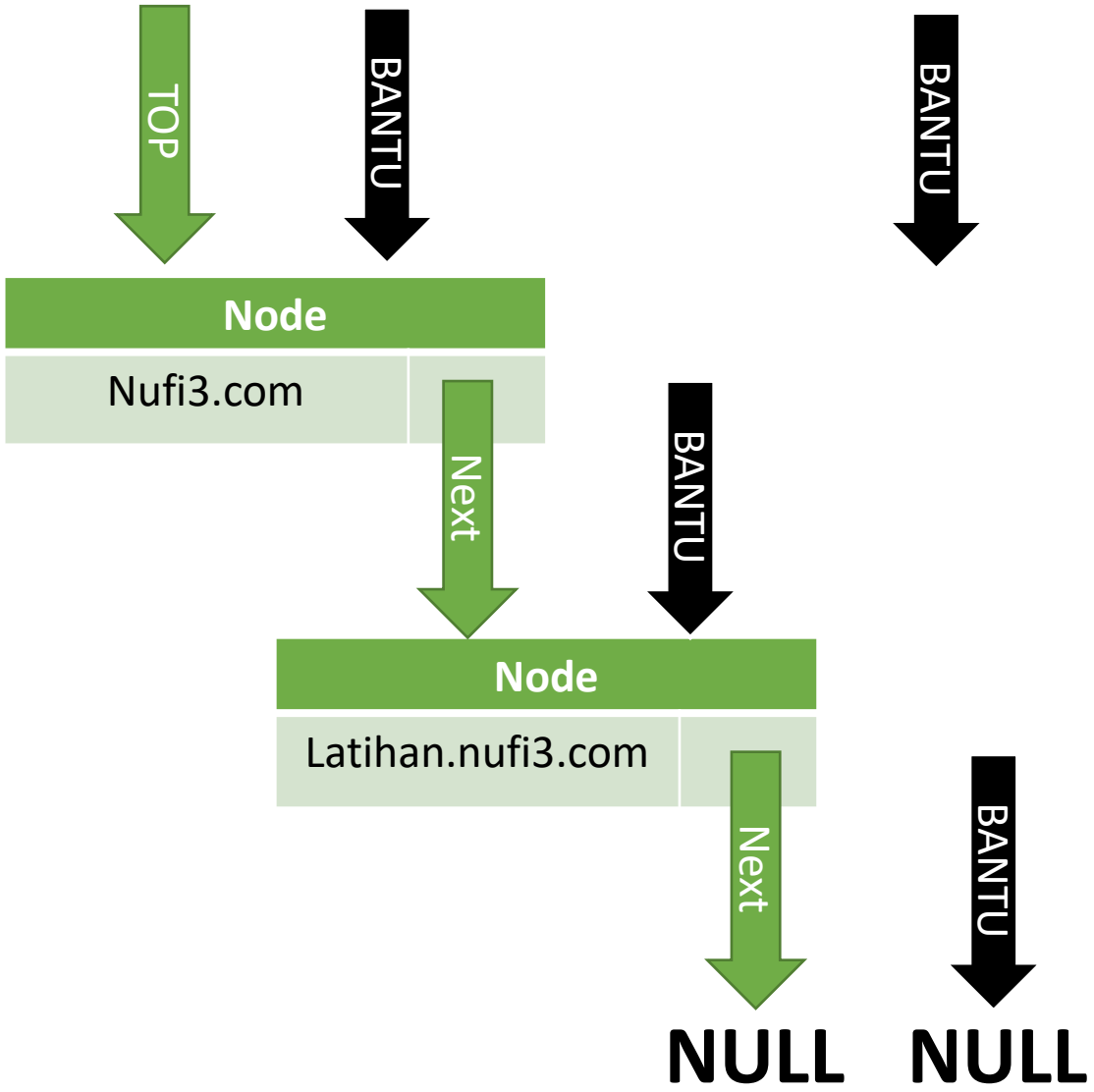

```
int hitungNode(){
    int hasil = 0;
    if(isKosong() == false){
        Node *bantu = new Node();
        bantu = top;

        while(bantu != NULL){
            hasil += 1;

            bantu = bantu->next;
        }
    }

    return hasil;
}
```

Operasi Pendukung Stack




```
int hitungNode(){
    int hasil = 0;
    if(isKosong() == false){
        Node *bantu = new Node();
        bantu = top;

        while(bantu != NULL){
            hasil += 1;

            bantu = bantu->next;
        }
    }

    return hasil;
}
```

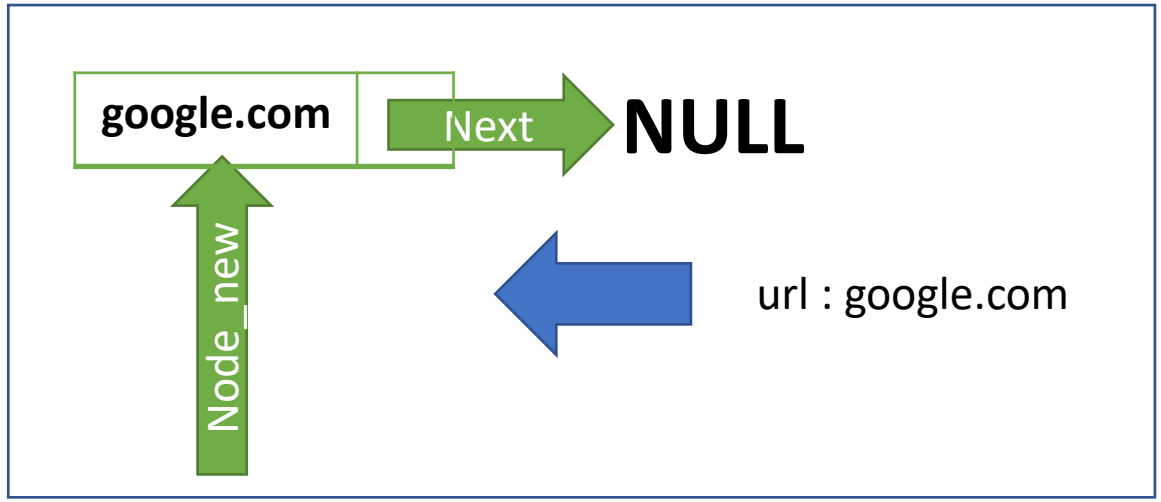
Operasi Pendukung Stack




```
72 void cetakIsiStack(){
73     if(isKosong() == false){
74         cout<<"Isi Stack ["<<hitungNode()<<"] : "<< endl;
75 
76         Node *bantu = top;
77         int i = 1;
78         while(bantu != NULL){
79             cout<<bantu->url<<" | ";
80 
81             bantu = bantu->next;
82             i++;
83         }
84         cout<<"\n";
85     }else{
86         cout<<"Stack kosong"<<endl;
87     }
88 }
89 
```

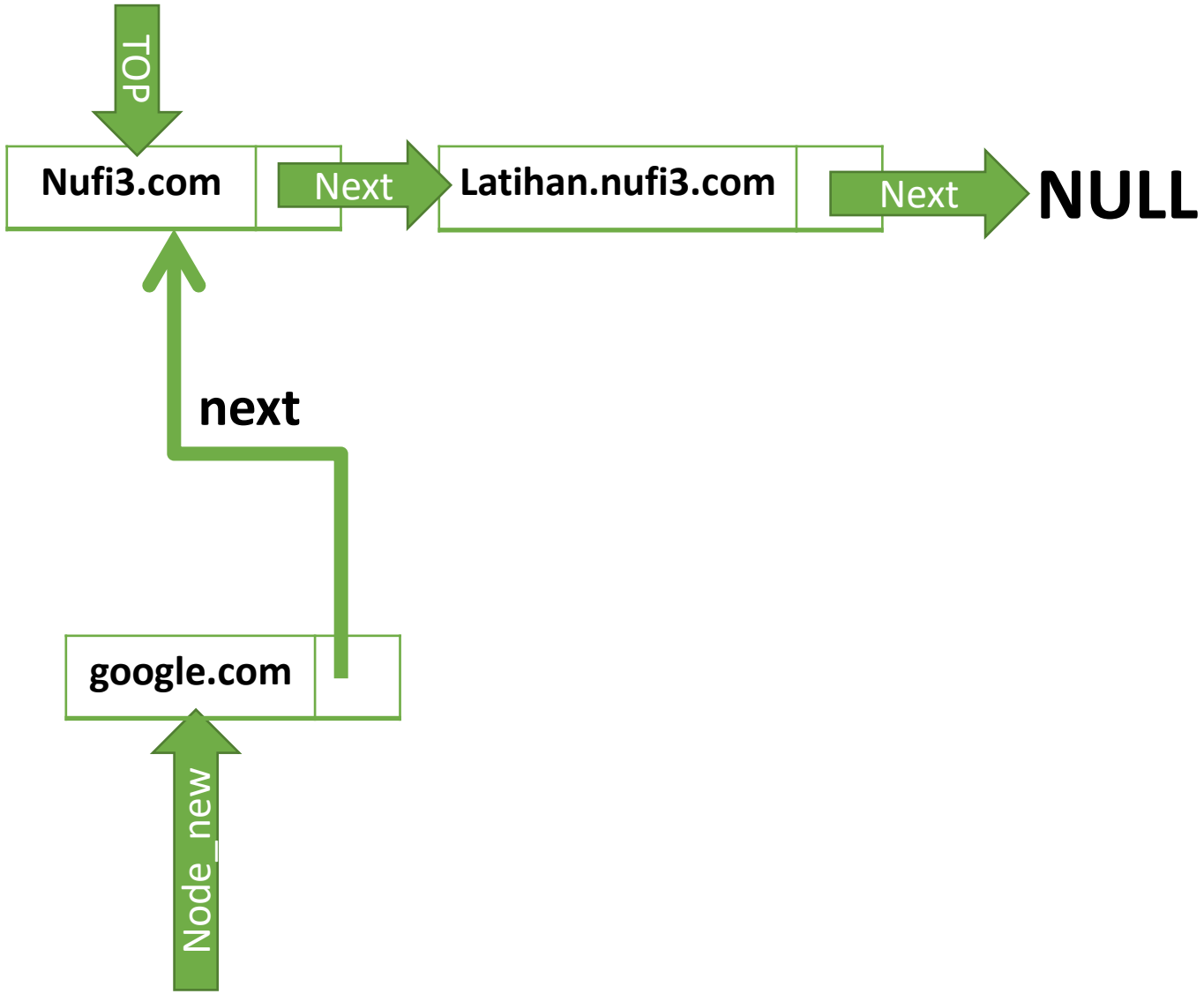
Push : Add New Elemen

```
int pushNode(string url_new){
    Node *node_new = new Node();
    node_new->url = url_new;

    if(isKosong() == true){
        node_new->next = NULL;
    }else{
        node_new->next = top;
    }

    top = node_new;
    node_new = NULL;

    return 0;
}
```



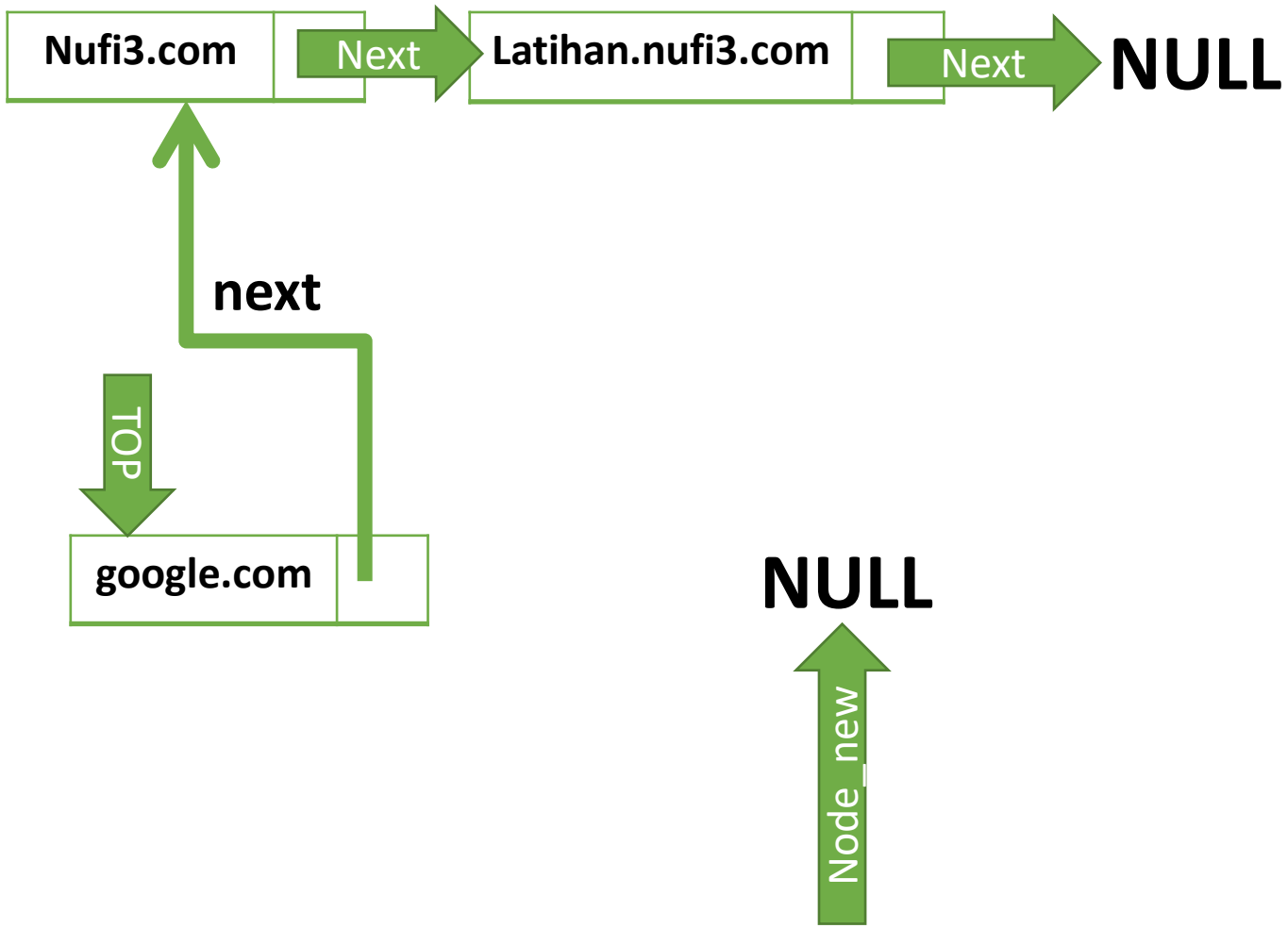
Push : Add New Elemen

```
int pushNode(string url_new){
    Node *node_new = new Node();
    node_new->url = url_new;

    if(isKosong() == true){
        node_new->next = NULL;
    }else{
        node_new->next = top;
    }

    top = node_new;
    node_new = NULL;

    return 0;
}
```



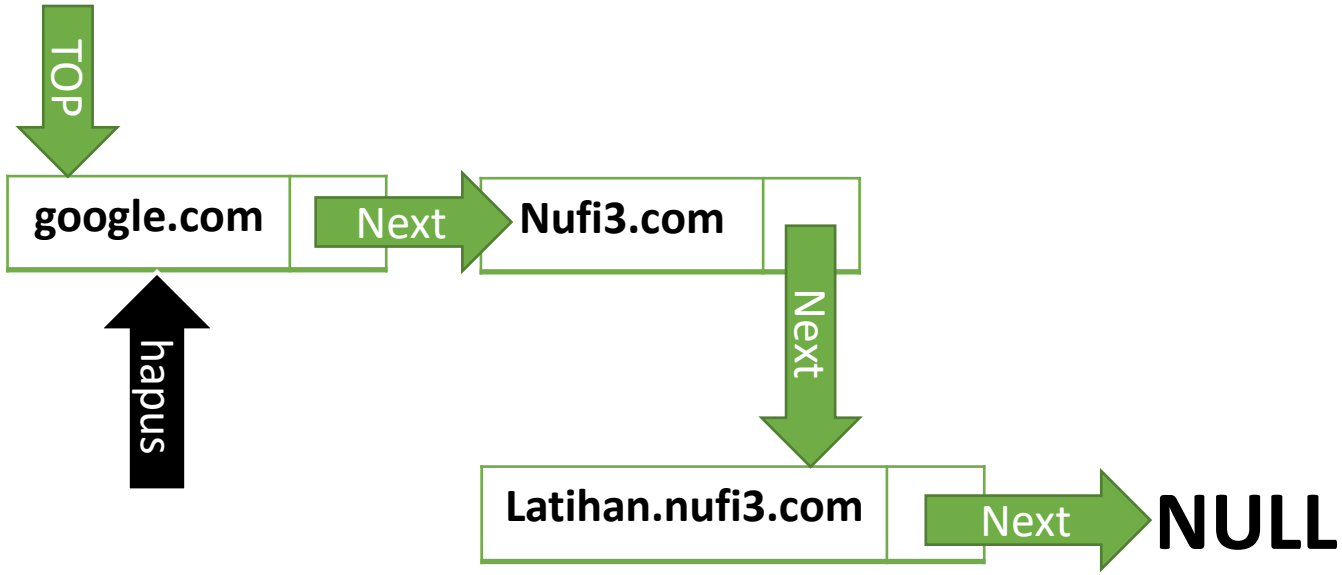
Push : Add New Elemen

```
int pushNode(string url_new){
    Node *node_new = new Node();
    node_new->url = url_new;

    if(isKosong() == true){
        node_new->next = NULL;
    }else{
        node_new->next = top;
    }

    top = node_new;
    node_new = NULL;

    return 0;
}
```



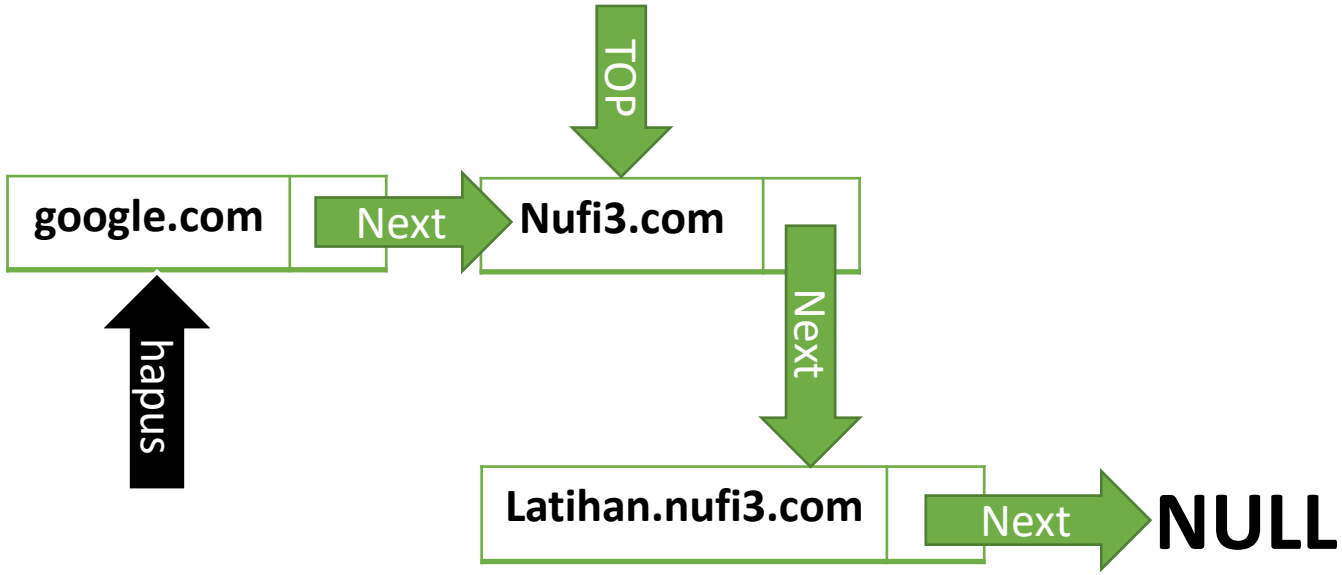
Pop : Delete Head Node

```
53 int popNode(){
54     if(isKosong() == false){
55         Node* hapus = top;
56 
57         if(hitungNode() == 1){
58             top = NULL;
59         }else{
60             top = top->next;
61         }
62 
63         hapus->next = NULL;
64         free(hapus);
65     }else{
66         cout << "stack kosong" << endl;
67     }
68 
69     return 0;
70 }
```



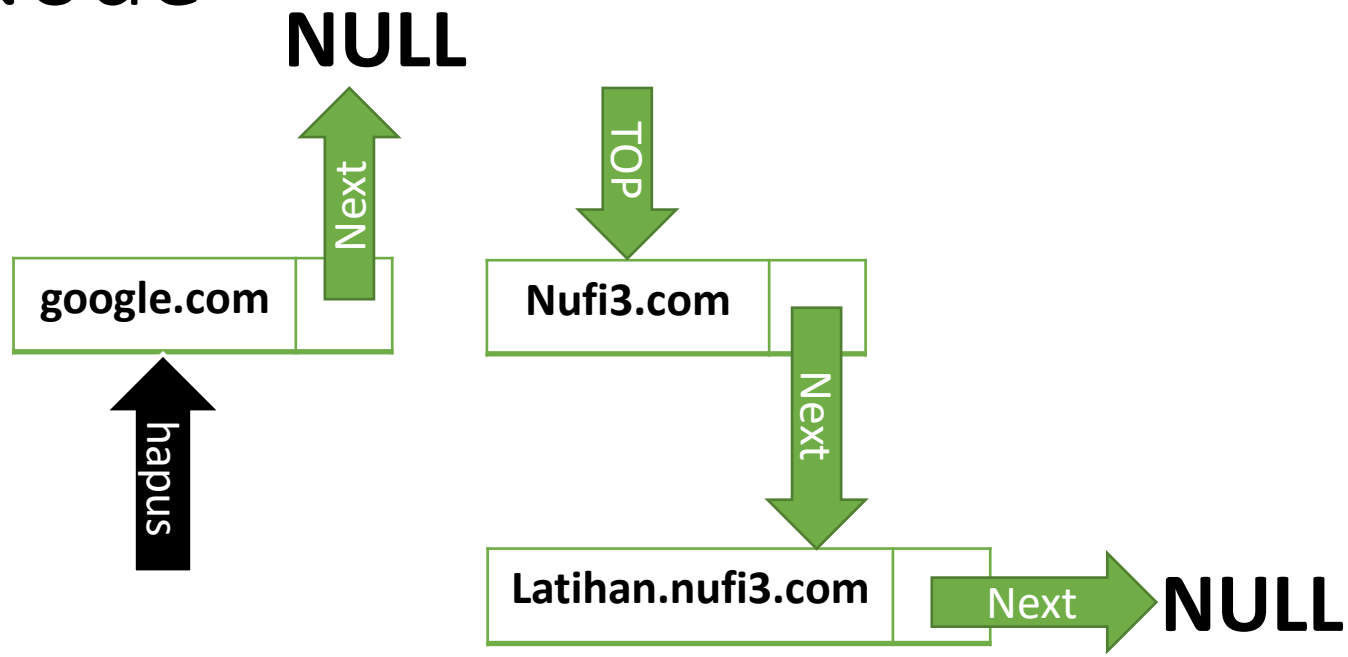
Pop : Delete Head Node

```
53 int popNode(){
54     if(isKosong() == false){
55         Node* hapus = top;
56 
57         if(hitungNode() == 1){
58             top = NULL;
59         }else{
60             top = top->next;
61         }
62 
63         hapus->next = NULL;
64         free(hapus);
65     }else{
66         cout << "stack kosong" << endl;
67     }
68 
69     return 0;
70 }
```



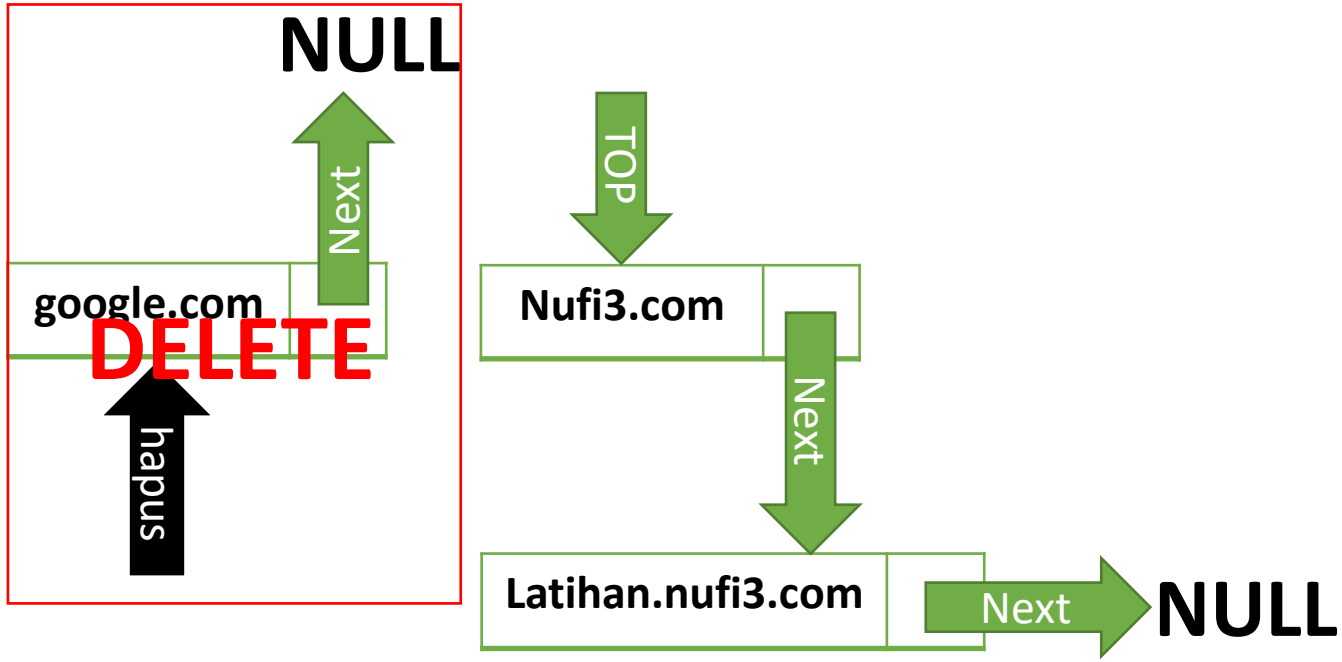
Pop : Delete Head Node

```
53 int popNode(){
54     if(isKosong() == false){
55         Node* hapus = top;
56 
57         if(hitungNode() == 1){
58             top = NULL;
59         }else{
60             top = top->next;
61         }
62 
63         hapus->next = NULL;
64         free(hapus);
65     }else{
66         cout << "stack kosong" << endl;
67     }
68 
69     return 0;
70 }
```



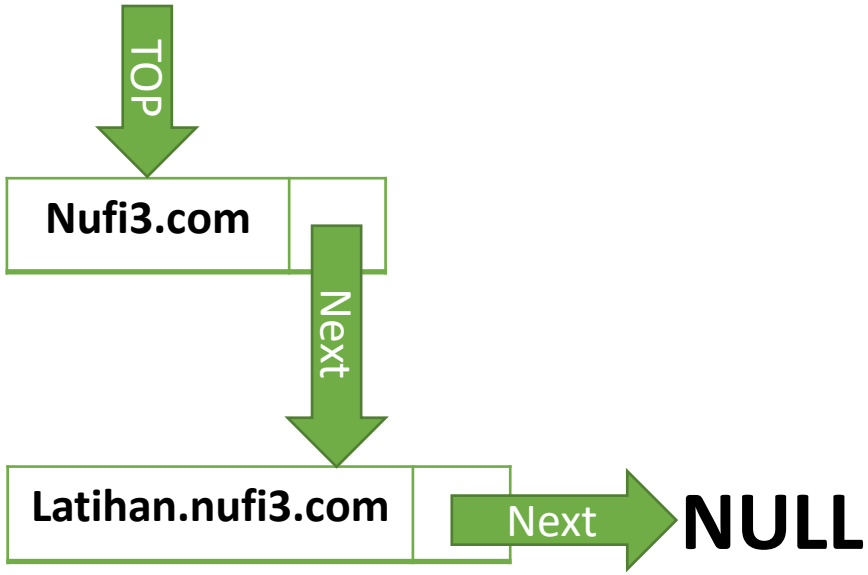
Pop : Delete Head Node

```
53 int popNode(){
54     if(isKosong() == false){
55         Node* hapus = top;
56
57         if(hitungNode() == 1){
58             top = NULL;
59         }else{
60             top = top->next;
61         }
62
63         hapus->next = NULL;
64         free(hapus);
65     }else{
66         cout << "stack kosong" << endl;
67     }
68
69     return 0;
70 }
```



Pop : Delete Head Node

```
53 int popNode(){
54     if(isKosong() == false){
55         Node* hapus = top;
56 
57         if(hitungNode() == 1){
58             top = NULL;
59         }else{
60             top = top->next;
61         }
62 
63         hapus->next = NULL;
64         free(hapus);
65     }else{
66         cout << "stack kosong" << endl;
67     }
68 
69     return 0;
70 }
```



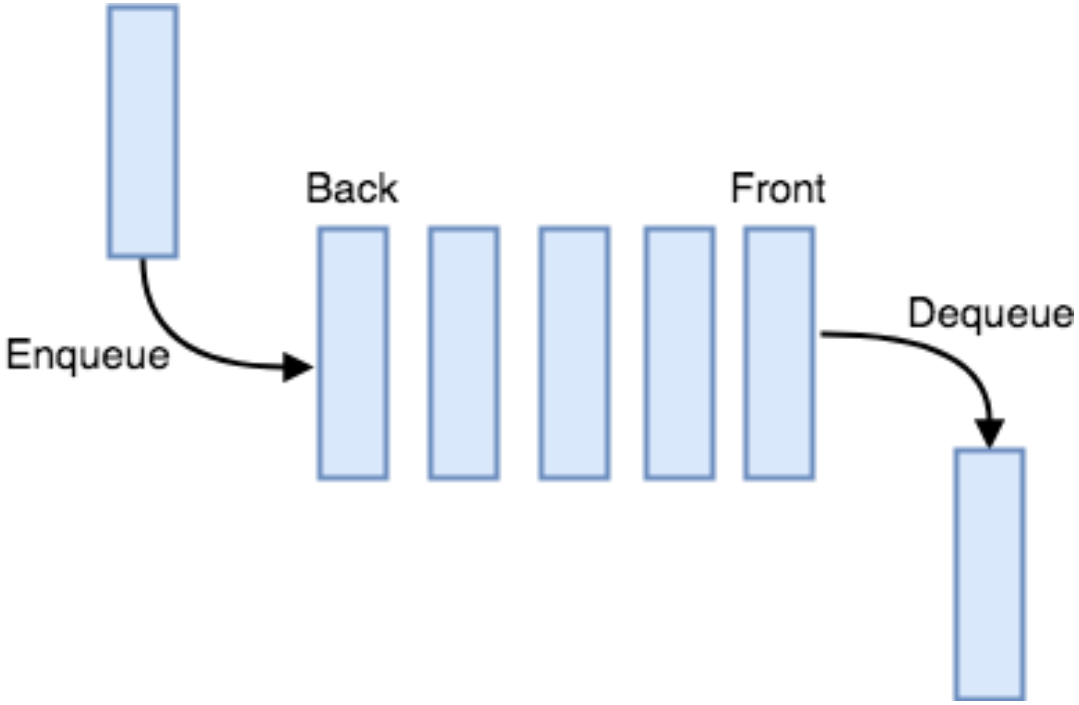
Main Program

```
90  int main(){
91      top = new Node();
92
93      cout<<"mulai\n";
94      buatKosong();
95      cetakIsiStack();
96
97      cout<<"Masukan 3 URL histori browser \n";
98      pushNode("http://latihan.nufi3.com");
99      pushNode("http://nufi3.com");
100     pushNode("http://www.stmik-tasikmalaya.ac.id");
101     cetakIsiStack();
102
103     cout<<"Hapus 1 Node paling atas\n";
104     popNode();
105     cetakIsiStack();
106
107
108     return 0;
109 }
```

input

```
mulai
Stack kosong
Masukan 3 URL histori browser
Isi Stack [3] :
http://www.stmik-tasikmalaya.ac.id | http://nufi3.com | http://latihan.nufi3.com |
Hapus 1 Node paling atas
Isi Stack [2] :
http://nufi3.com | http://latihan.nufi3.com |


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

Sekarang, bagaimana Queue diimplementasikan pada Linked List?



Operasi pada Queue

- Buat kosong : first = last = -1
- Cek apakah kosong : first == -1? True or false
- Cek apakah penuh : last == (N-1)? True or false
- Informasi Posisi Last : return last (indeks)
- Cetak isi Queue : membaca dari First ke Last
- Tambah : menambah elemen pada bagian belakang/ akhir, Last++
- Hapus : menghapus elemen pada bagian depan/ awal, Elemen bergeser ke depan, Last--

Operasi pada Queue

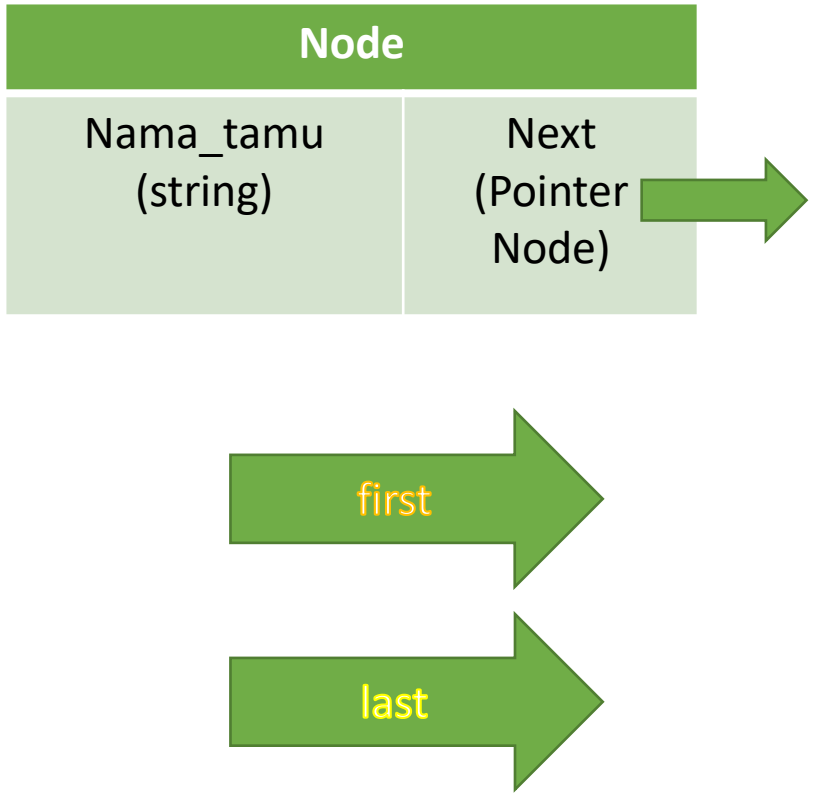
- Buat kosong : first = last = -1
- Cek apakah kosong : first == -1? True or false
- Cek apakah penuh : last == (N-1)? True or false
- Informasi Posisi Last : return last (indeks)
- Cetak isi Queue : membaca dari First ke Last
- Tambah : menambah elemen pada bagian belakang/ akhir, Last++
- Hapus : menghapus elemen pada bagian depan/ awal, Elemen bergeser ke depan, Last--

Operasi pada Queue menggunakan Linked List

- Buat kosong : first = last = -1 → first = last = NULL
- Cek apakah kosong : first == -1? → first == NULL → True or false
- Cek apakah penuh : last == (N-1)? True or false → **TIDAK ADA**
- Informasi Posisi Last : return last (indeks) → isi Node **Last/ Tail**
- Cetak isi Queue : membaca dari Node First sampe Node Last
- Tambah : menambah elemen pada bagian belakang/ akhir, Last++
  - Menambahkan node pada bagian akhir/ LAST/ TAIL
- Hapus : menghapus elemen pada bagian depan/ awal, Elemen bergeser ke depan, Last--
  - Menghapus node pada bagian depan/ Head/ Posisi 0

Implementasi pada C++

```
6  class Node{
7      public:
8          string nama_tamu;
9          Node* next;
10 };
11
12 Node* first;
13 Node* last;
```

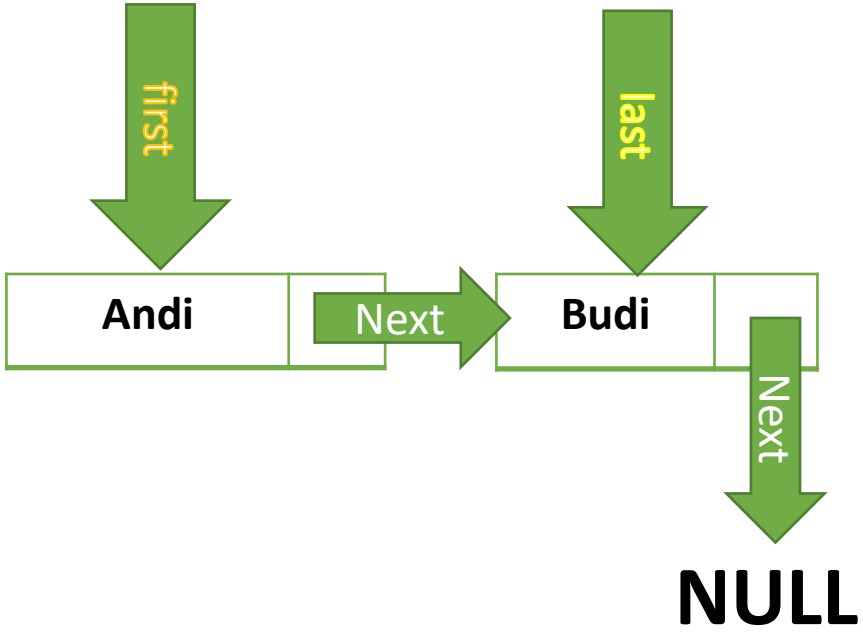


Operasi Pendukung Queue

```
15 void buatKosong(){
16     first = NULL;
17     last= NULL;
18 }
19
20 bool isKosong(){
21     return (first == NULL);
22 }
```

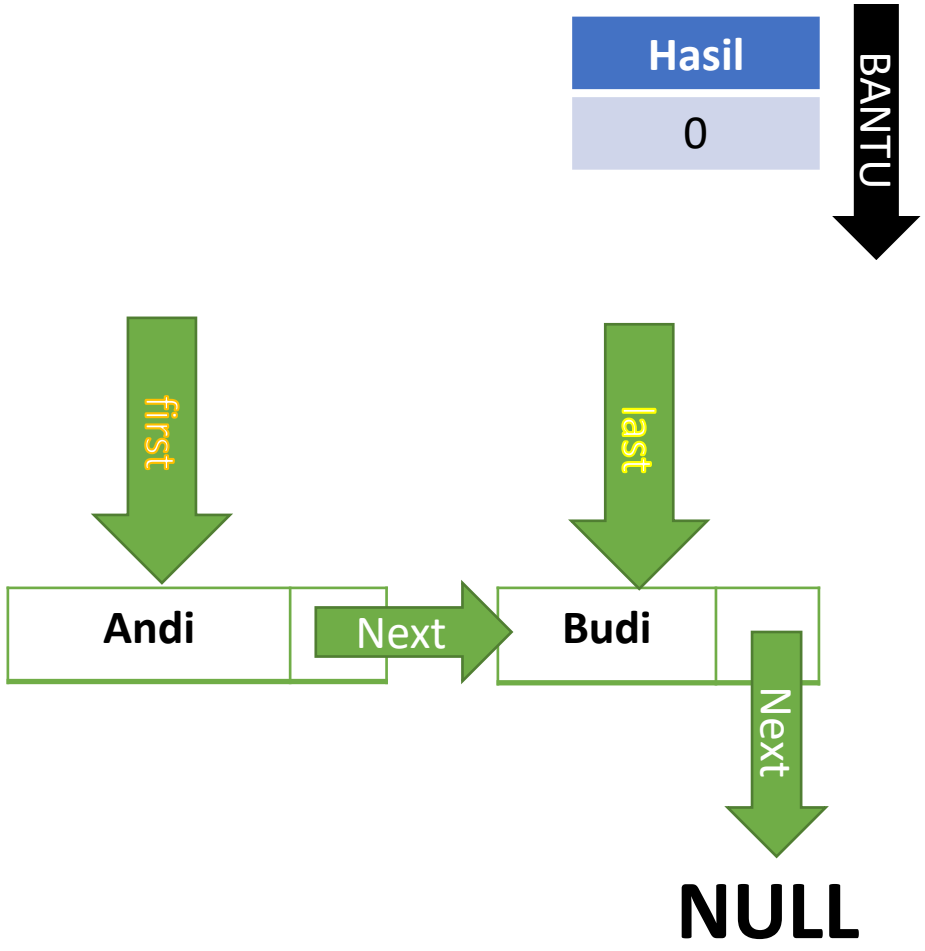
Tidak mengarah pada objek Node

KOSONG



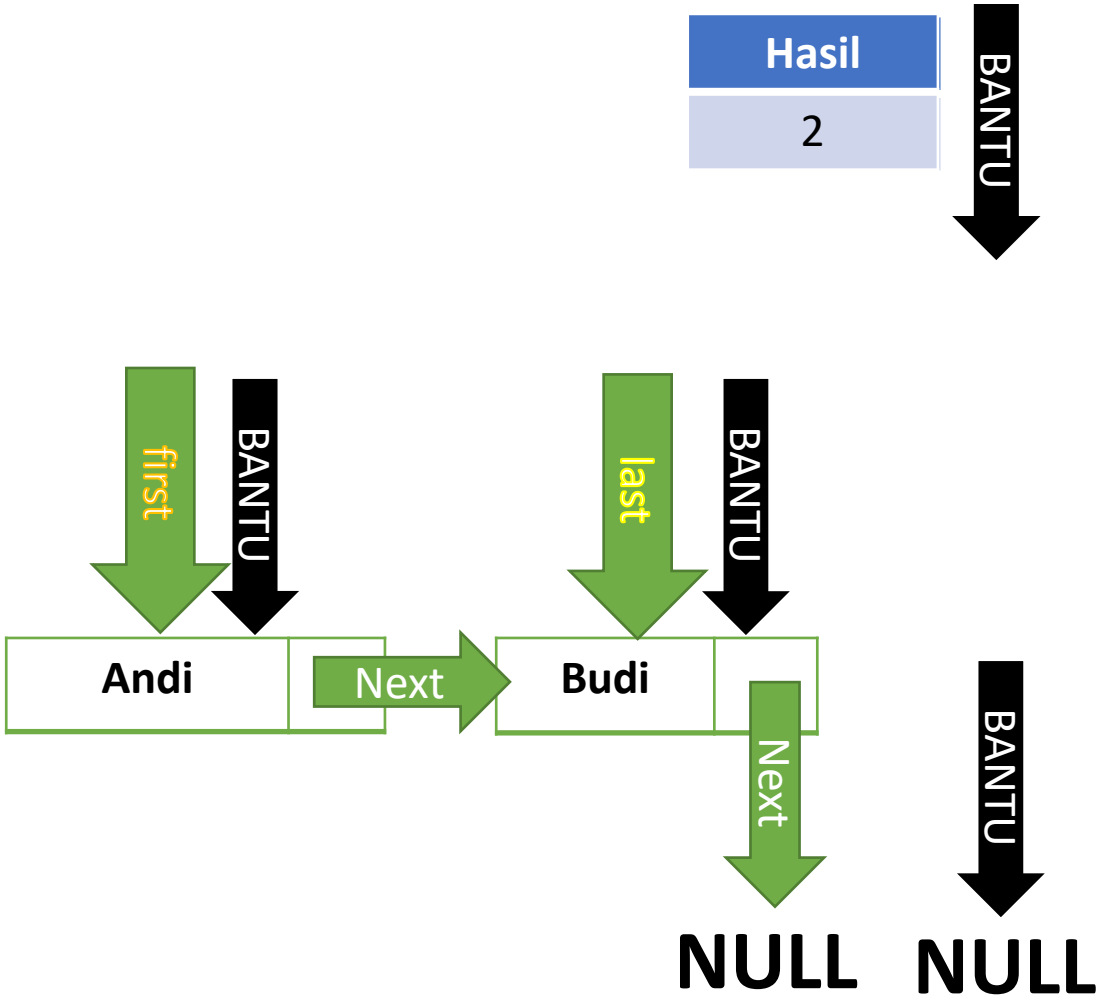
TIDAK KOSONG

Operasi Pendukung Queue



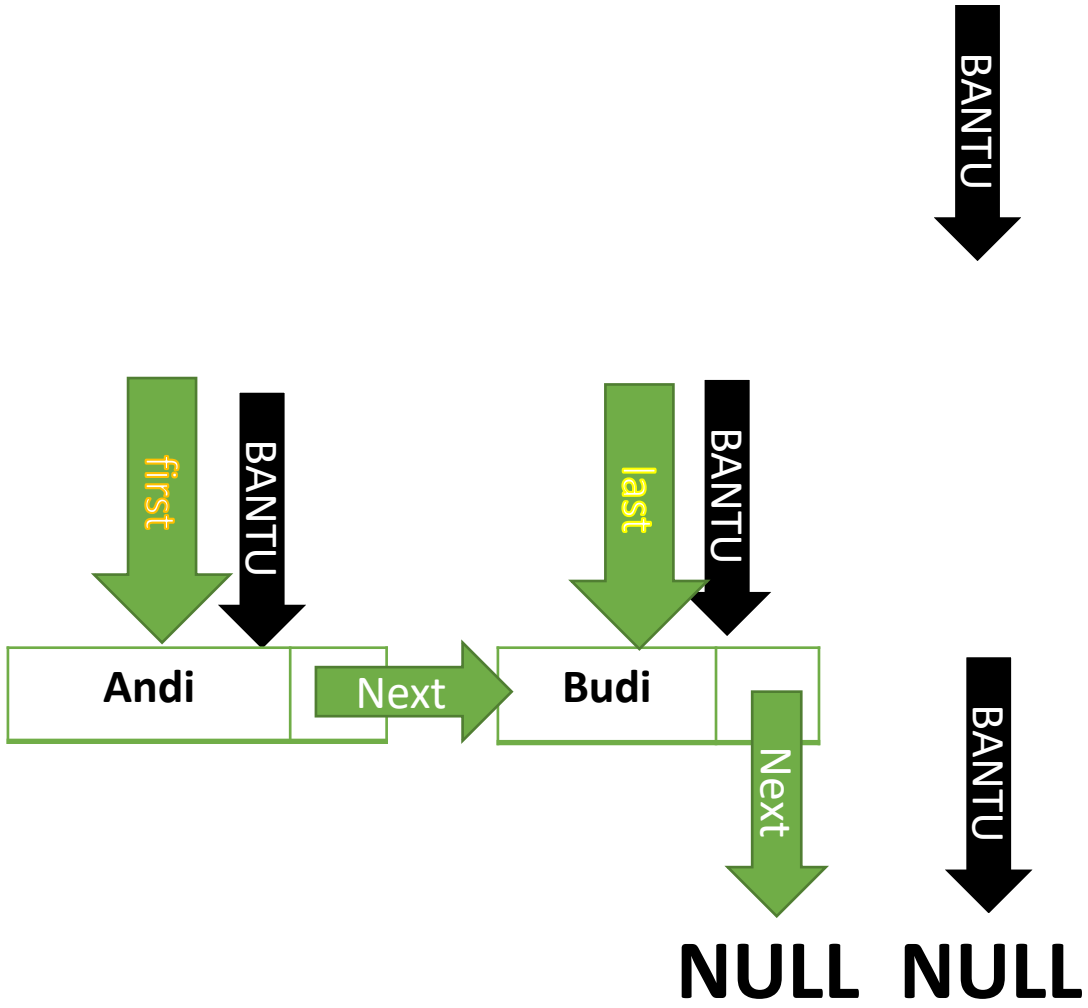

```
24 int hitungJumlahNode(){
25     int jumlah = 0;
26     if(isKosong() == false){
27         Node* bantu = first;
28         while(bantu != NULL){
29             jumlah++;
30
31             bantu = bantu->next;
32         }
33     }
34
35     return jumlah;
36 }
```

Operasi Pendukung Stack




```
int hitungJumlahNode(){
    int jumlah = 0;
    if(isKosong() == false){
        Node* bantu = first;
        while(bantu != NULL){
            jumlah++;

            bantu = bantu->next;
        }
    }

    return jumlah;
}
```

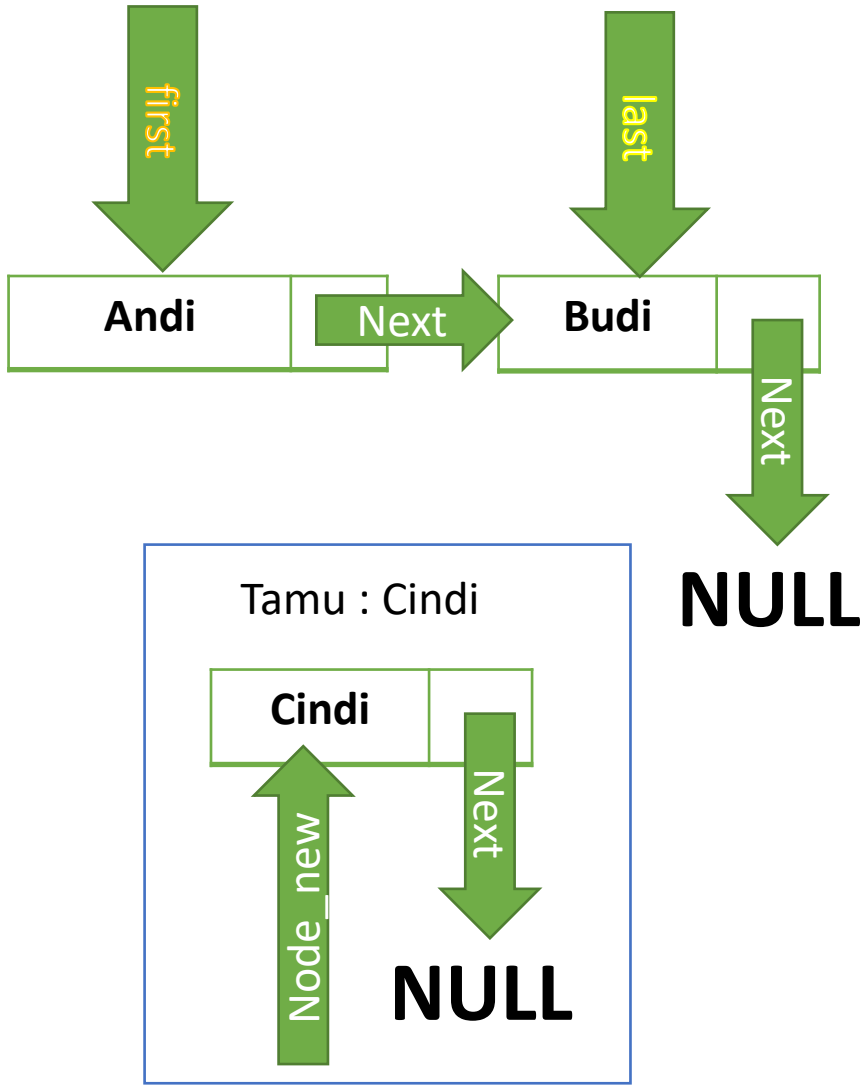
Operasi Pendukung Stack




```
38 void cetakIsiQueue(){
39     cout << "Daftar Antrian [" <<hitungJumlahNode()<<"] = "<< endl;
40     if(isKosong() == false){
41         Node *bantu = first;
42         int i = 1;
43         while(bantu != NULL){
44             cout<<bantu->nama_tamu<<" | ";
45
46             bantu = bantu->next;
47             i++;
48         }
49         cout<<"\n";
50     }else{
51         cout<<"Antrian kosong"<<endl;
52     }
53 }
```

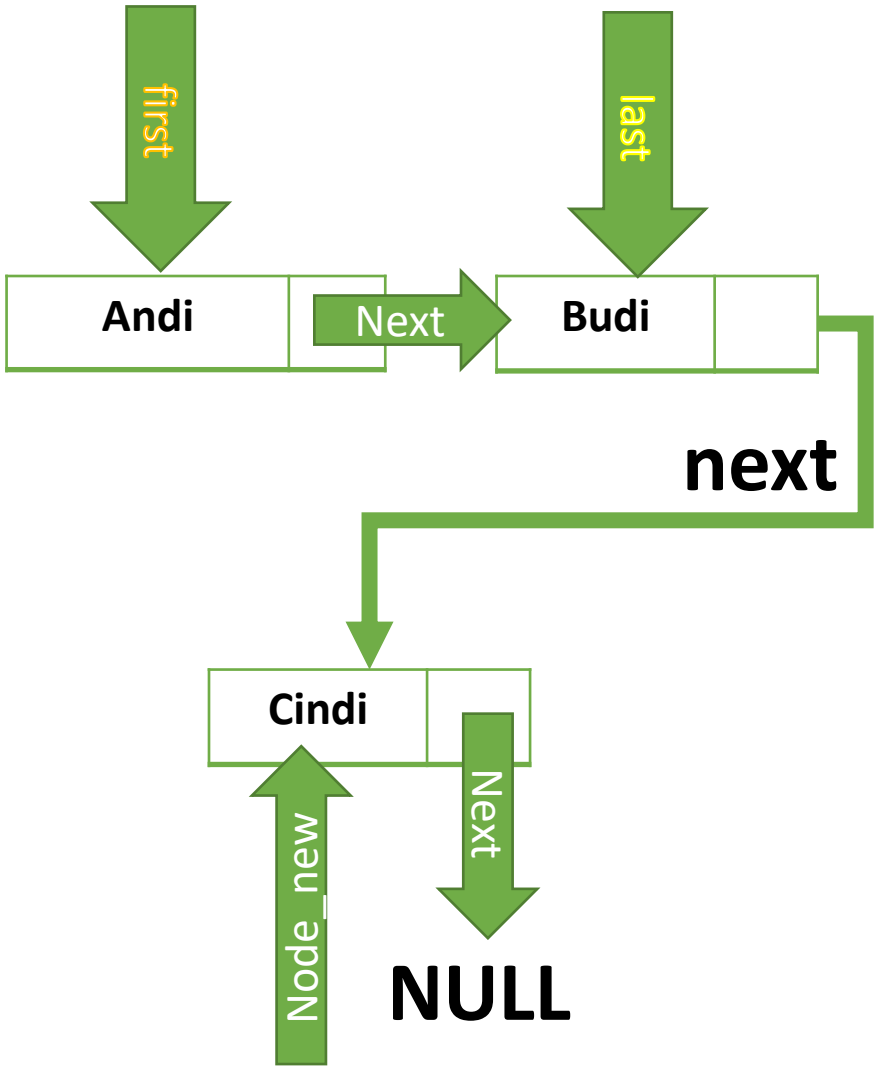
Tambah Node Baru di Bagian Akhir (Last)

```
56 void tambahNode(string tamu_baru){
57     Node * node_baru = new Node();
58     node_baru->nama_tamu = tamu_baru;
59     node_baru->next = NULL;
60
61     if(isKosong() == true){
62         first = last = node_baru;
63     }
64
65     last->next = node_baru;
66     last = node_baru;
67 }
```



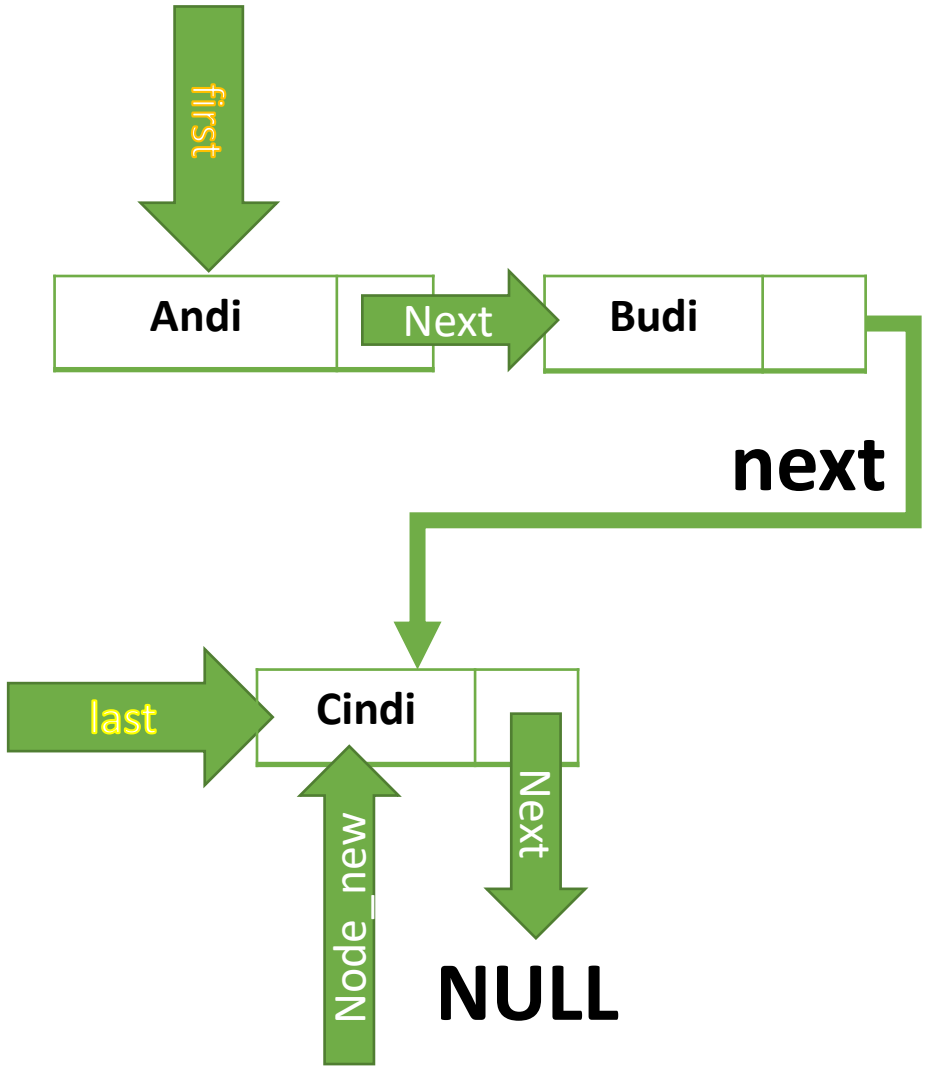
Tambah Node Baru di Bagian Akhir (Last)

```
56 void tambahNode(string tamu_baru){
57     Node * node_baru = new Node();
58     node_baru->nama_tamu = tamu_baru;
59     node_baru->next = NULL;
60
61     if(isKosong() == true){
62         first = last = node_baru;
63     }
64
65     last->next = node_baru;
66     last = node_baru;
67 }
```



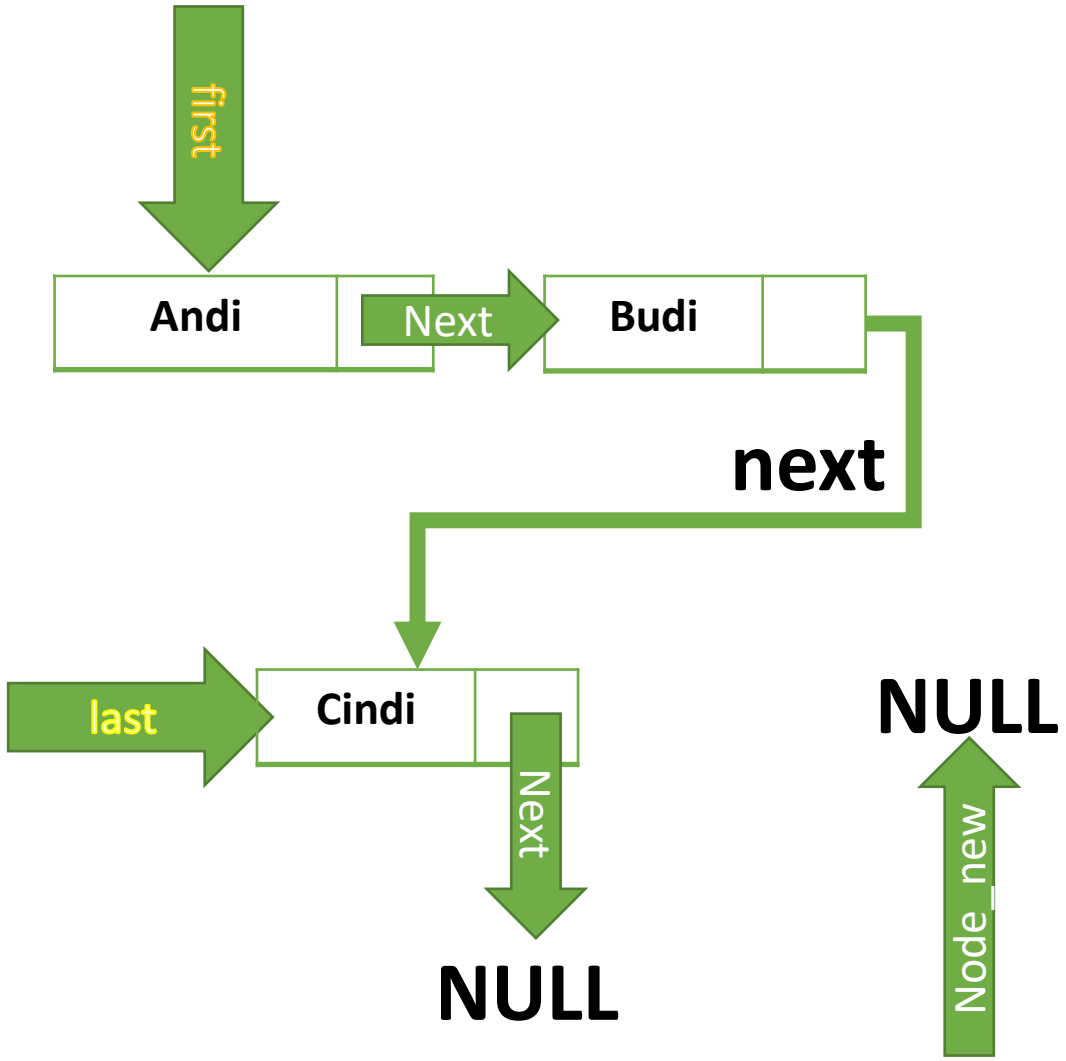
Tambah Node Baru di Bagian Akhir (Last)

```
56 void tambahNode(string tamu_baru){
57     Node * node_baru = new Node();
58     node_baru->nama_tamu = tamu_baru;
59     node_baru->next = NULL;
60
61     if(isKosong() == true){
62         first = last = node_baru;
63     }
64
65     last->next = node_baru;
66     last = node_baru;
67 }
```



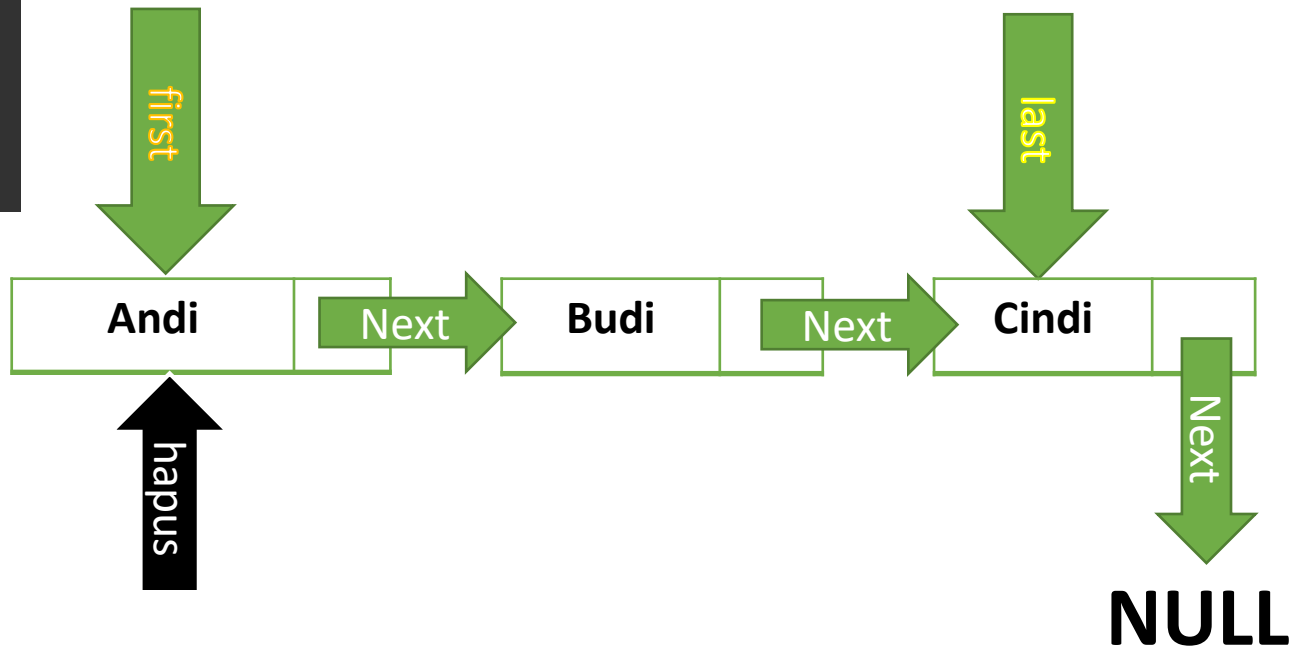
Tambah Node Baru di Bagian Akhir (Last)

```
56 void tambahNode(string tamu_baru){
57     Node * node_baru = new Node();
58     node_baru->nama_tamu = tamu_baru;
59     node_baru->next = NULL;
60
61     if(isKosong() == true){
62         first = last = node_baru;
63     }
64
65     last->next = node_baru;
66     last = node_baru;
67 }
```



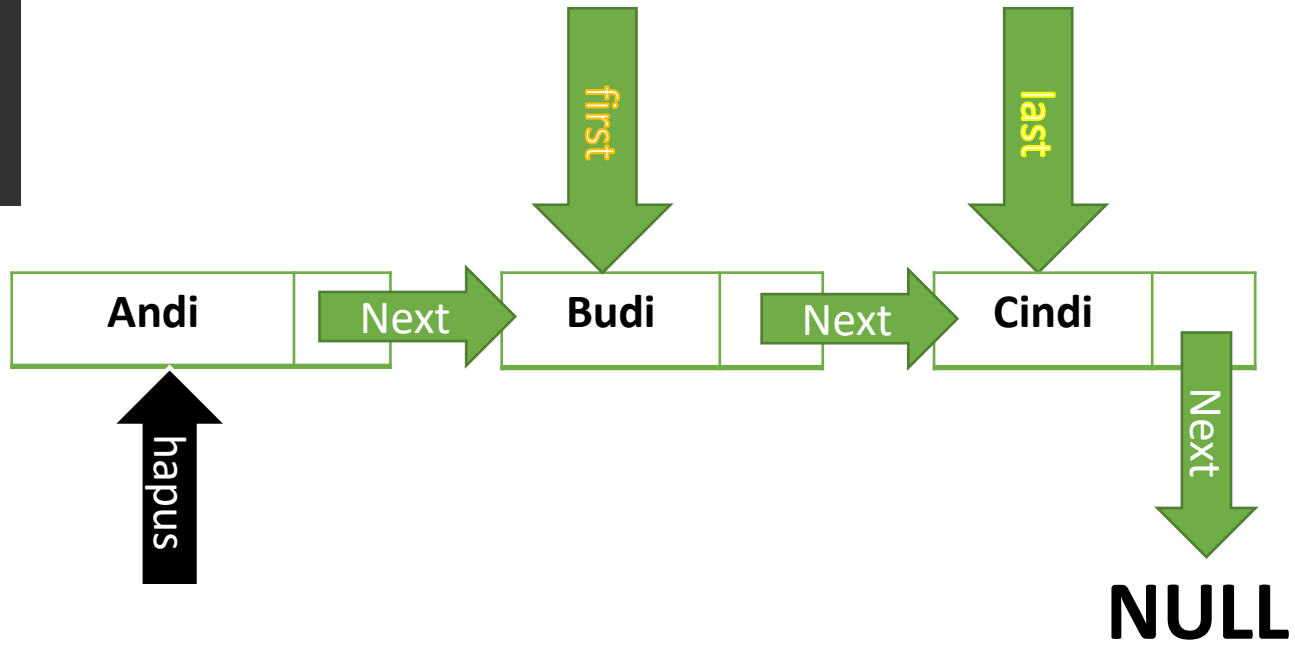
Hapus Node Paling Depan

```
69 void hapusNode(){
70     if(isKosong() == true) cout<<"Antrian kosong\n";
71
72     Node* node_hapus = first;
73     first = node_hapus->next;
74     free(node_hapus);
75 }
```



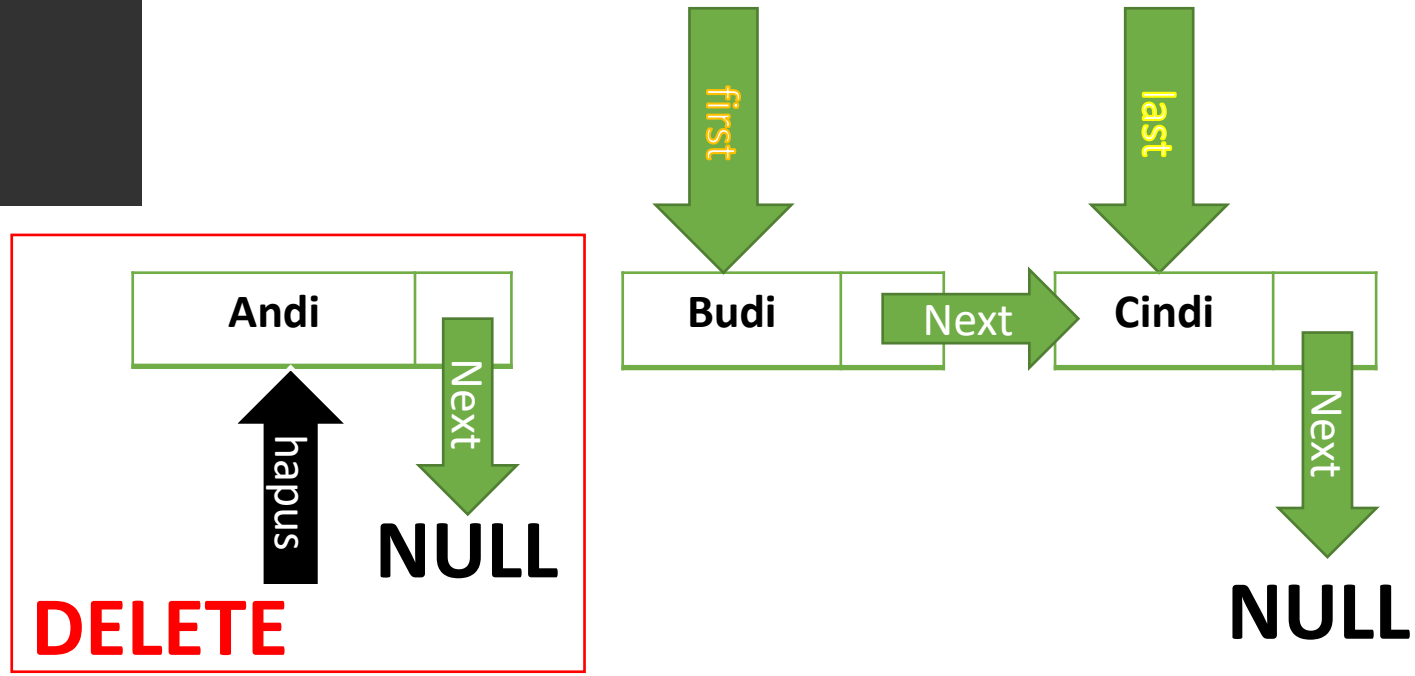
Hapus Node Paling Depan

```
69 void hapusNode(){
70     if(isKosong() == true) cout<<"Antrian kosong\n";
71
72     Node* node_hapus = first;
73     first = node_hapus->next;
74     free(node_hapus);
75 }
```



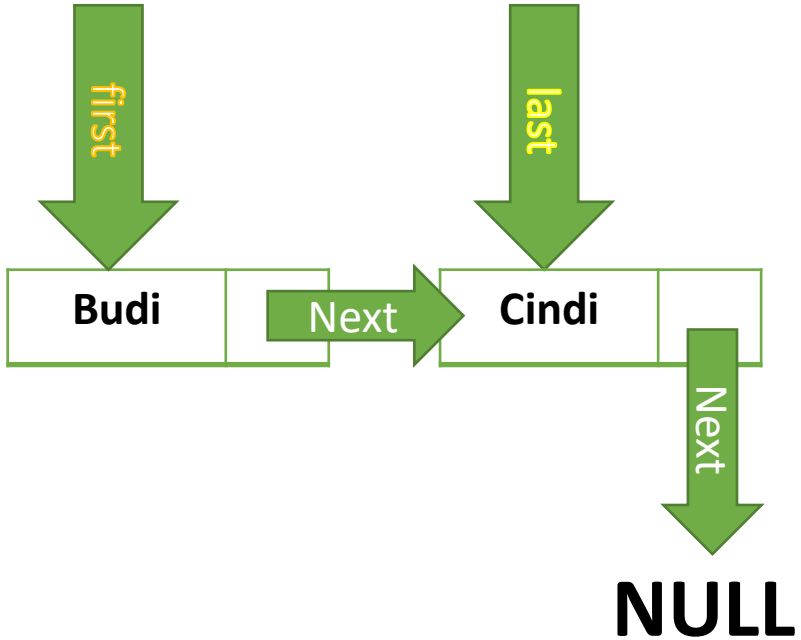
Hapus Node Paling Depan

```
69 void hapusNode(){
70     if(isKosong() == true) cout<<"Antrian kosong\n";
71
72     Node* node_hapus = first;
73     first = node_hapus->next;
74     free(node_hapus);
75 }
```



Hapus Node Paling Depan

```
69 void hapusNode(){
70     if(isKosong() == true) cout<<"Antrian kosong\n";
71
72     Node* node_hapus = first;
73     first = node_hapus->next;
74     free(node_hapus);
75 }
```



Main Program

```
77  int main(){
78      first = new Node();
79      last = new Node();
80
81      cout<<"mulai\n";
82      buatKosong();
83      cetakIsiQueue();
84
85      cout<<"Masukan 3 Nama Tamu \n";
86      tambahNode("Rahmat");
87      tambahNode("Udin");
88      tambahNode("Kokom");
89      cetakIsiQueue();
90
91      cout<<"Hapus 1 Node paling depan\n";
92      hapusNode();
93      cetakIsiQueue();
94
95
96      return 0;
97  }
```

```
mulai
Daftar Antrian [0] =
Antrian kosong
Masukan 3 Nama Tamu
Daftar Antrian [3] =
Rahmat | Udin | Kokom |
Hapus 1 Node paling depan
Daftar Antrian [2] =
Udin | Kokom |


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

Pertanyaan :

-Buka browser www.menti.com

- masukan kode 70 11 31

Go to www.menti.com and use the code 70 11 31

https://bit.ly/presensi3Juli2020

Link Rekap Kehadiran

Rewards Event Last Meeting

Proyek Akhir untuk Kelas E & F

Akan disampaikan di Grup Telegram

Terima Kasih