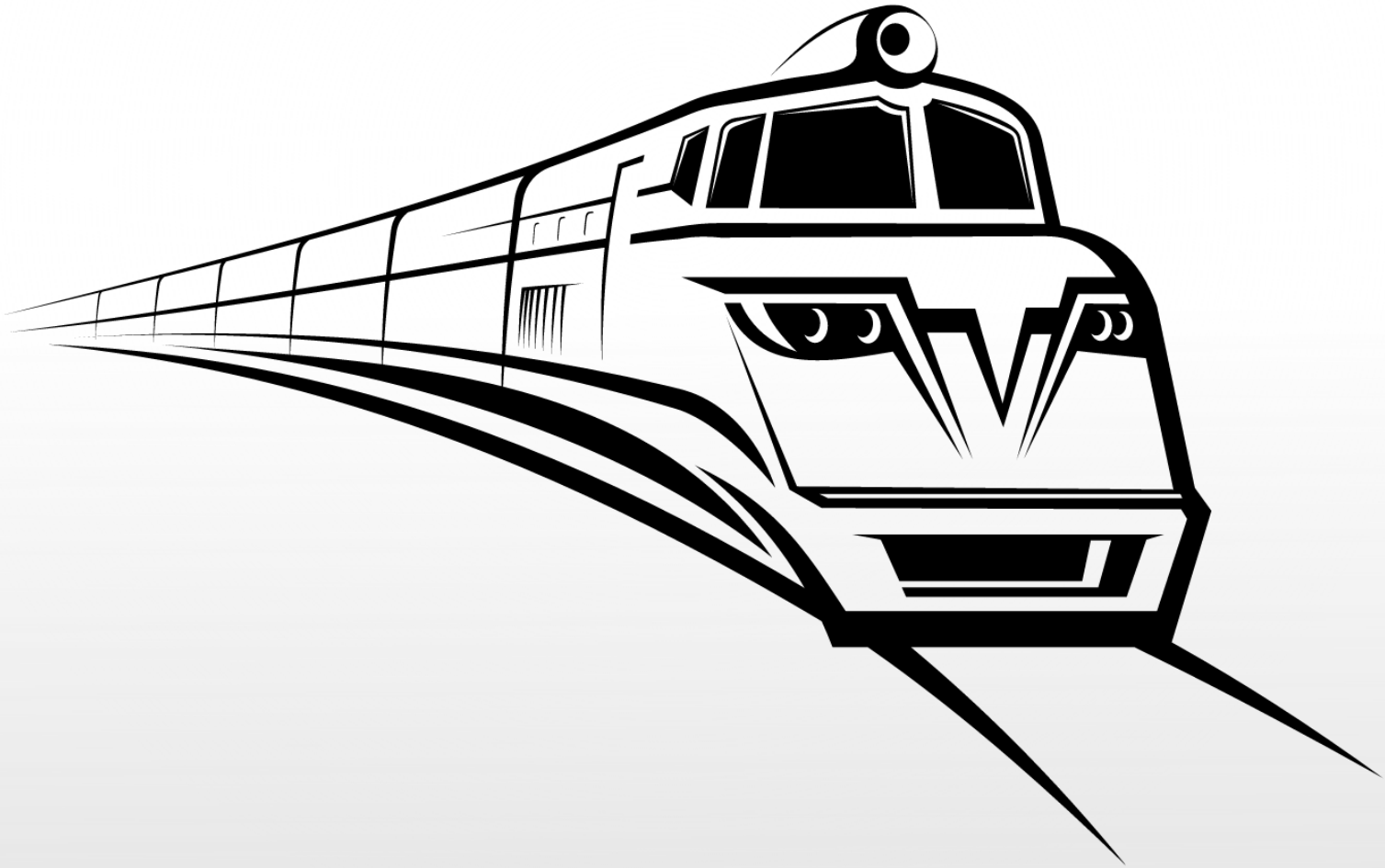


LINKED LIST

ASISTEN INFORMASI
DAN STRUKTUR DATA

Modul 6

Linked List 1

Tujuan

1. Praktikan dapat mengetahui manfaat dari penggunaan linked list
2. Praktikan dapat memahami bagaimana cara kerja dari linked list
3. Praktikan dapat menerapkan konsep linked list dalam program dan kehidupan sehari – hari

Latar Belakang

Linked list merupakan sebuah bentuk struktur data yang bersifat **dinamis**, yang tersusun atas node-node yang saling terhubung secara sekuensial dengan menggunakan bantuan pointer. Berikut beberapa keuntungan menggunakan konsep linked list

1. **Bersifat dinamis**
   Berbeda dengan array dimana sebelum program dijalankan kita harus menulis berapa banyak elemen data yang akan ditampung pada array tersebut, dengan menggunakan linked list kita tidak perlu menuliskan hal tersebut.

2. **Hemat memori**
   Merujuk pada sifat pertama (**dinamis**), linked list tidak membutuhkan banyak memori seperti array, karena banyak atau tidaknya suatu node dalam linked list akan selalu bergantung kepada banyaknya data yang ingin disimpan oleh user tersebut, maka dari itu sifat dari linked list adalah **dinamis.**

Dasar Teori

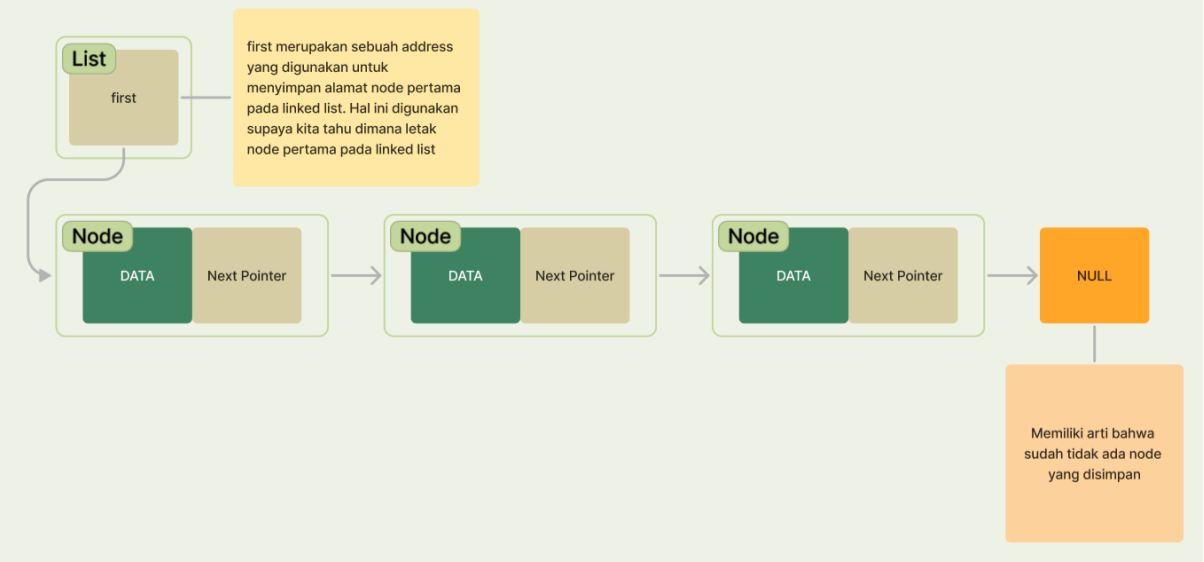
Struktur bagan Linked List

Terdiri dari dua bagian penting :

1. **Data**
   Data merupakan elemen dalam Linked List yang digunakan untuk menyimpan sebuah data. Elemen yang disimpan dapat berupa tipe data primitif (int, float) maupun tipe data bentukan (Mahasiswa, Dosen).
2. **Next Pointer**
   Next pointer merupakan elemen dalam Linked List yang digunakan untuk menyimpan alamat node selanjutnya.

   Berikut gambaran **data** dan **next pointer** dalam ilustrasi dan juga code

Ilustrasi Linked List yang memiliki 3 buah node



Untuk detail dari setiap bagian dan juga beserta codenya dapat dilihat sebagai berikut

DATA


```
typedef struct Node *address;

typedef struct Node{
    int angka;
    address next;
} tNode;
```

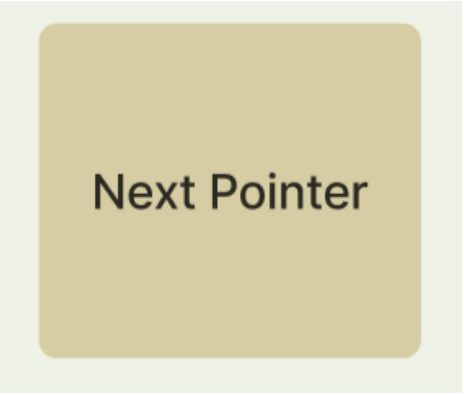
Data dengan tipe data primitif (integer)

```
typedef struct Node *address;

typedef struct {
    string nama;
    int umur;
}Mahasiswa;

typedef struct Node{
    Mahasiswa M;
    address next;
} tNode;
```

Data dengan tipe data bentukan (Mahasiswa)

Dapat dilihat pada gambar diatas, bahwa DATA yang disimpan dapat berupa tipe data primitif maupun bentukan.

Next Pointer




```
typedef struct Node *address;
```

Pembentukan variabel address yang merupakan tipe data bentukan yang bernama Node

```
typedef struct Node *address;

typedef struct {
    string nama;
    int umur;
}Mahasiswa;

typedef struct Node{
    int angka;
    address next;
} tNode;

typedef struct{
    address first;
} List;
```

Dapat dilihat pada gambar diatas bahwa *address*/Next Pointer digunakan untuk menyimpan alamat node selanjutnya.

Guided

Note : Pembahasan guided terdapat pada bagian akhir modul.

Inti guided: Menampung tipe data Dosen menggunakan konsep linked list, dan dapat melakukan beberapa fungsi seperti :

1. Insert First (Penambahan depan)
2. Insert Last (Penambahan belakang)
3. Delete First (Penghapusan depan)
4. Delete Last (Penghapusan belakang)
5. Show List (Menampilkan semua node di dalam list tersebut)

Header.h

```h
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

typedef char string [50];

typedef struct Node *address;

typedef struct {
    string nama;
    int NPP;
} Data;

typedef struct Node{
    Data D;
    address next;
} tNode;

typedef struct{
    address first;
} List;

void createEmpty(List *L);
bool isEmpty(List L);

address alokasiData(Data D);
Data makeData(string nama, int NPP);

void delokasiData(address P);

void insertFirst(List *L, Data D);
void insertLast(List *L, Data D);

void deleteFirst(List *L);
void deleteLast(List *L);

void showList(List L);
```

Source.c

```c
#include "header.h"

void createEmpty(List *L){
    L->first = NULL;
}

bool isEmpty(List L){
    return L.first == NULL;
}

address alokasiData(Data D){
    address p;

    p = (tNode *) malloc(sizeof(tNode));
    p->D = D;
    p->next = NULL;

    return p;
}

Data makeData(string nama, int NPP){
    Data temp;
    strcpy(temp.nama,nama);
    temp.NPP = NPP;
    return temp;
}

void delokasiData(address P){
    free(P);
}

void insertFirst(List *L, Data D){

    address P;

    P = alokasiData(D);
    P->next = L->first;

    L->first = P;

    printf("\n[!] Insert First [!]");

}

void insertLast(List *L, Data D){

    address P, last;

    if(isEmpty(*L))
        insertFirst(&(*L),D);
    else{

        last = L->first;
        P = alokasiData(D);

        while(last->next != NULL){
            last = last->next;
        }

        last->next = P;
        printf("\n[!] Insert Last [!]");

    }
}
```

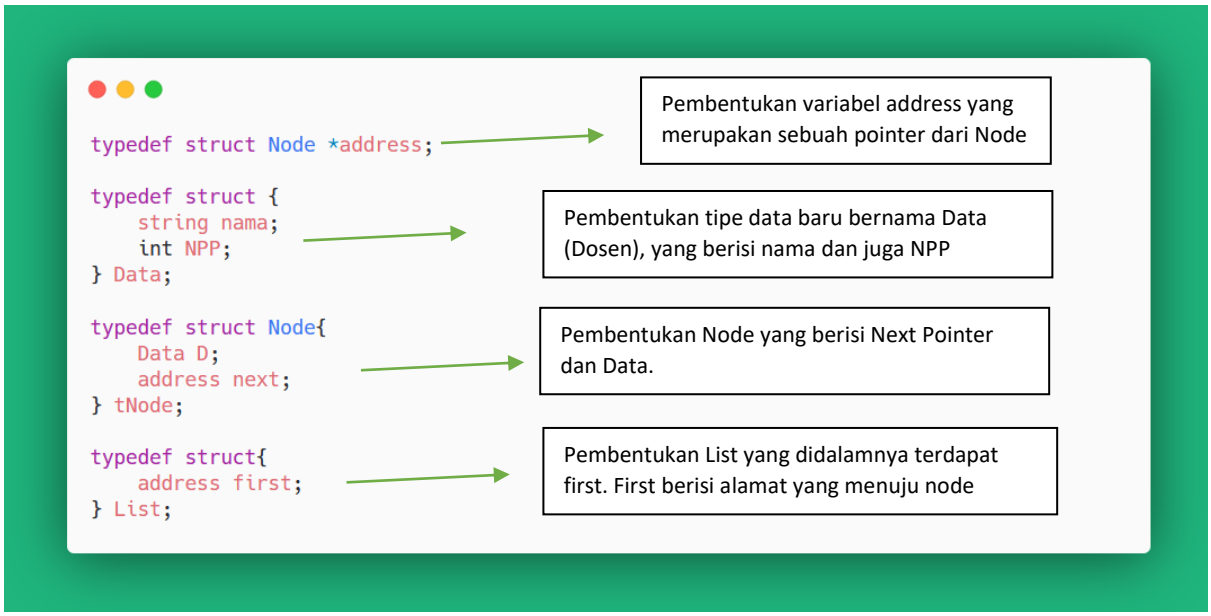
```c
void deleteFirst(List *L){
    address hapus;

    if(isEmpty(*L))
        printf("\n[!] List Empty [!]");
    else{

        hapus = L->first;
        L->first = L->first->next;
        free(hapus);
        printf("\n[!] Deleted [!]");

    }
}
void deleteLast(List *L){
    address last, hapus;
    if(isEmpty(*L))
        printf("\n[!] List Empty [!]");
    else
    {
        if(L->first->next == NULL)
        {
            deleteFirst(&(*L));
        }
        else
        {
            last = L->first;
            while(last->next->next != NULL)
            {
                last = last->next;
            }

            hapus = last->next;
            last->next = NULL;
            free(hapus);

            printf("[!] Deleted [!]");
        }
    }

}

void showList(List L){

    address P;

    if (isEmpty(L))
    {
        printf("\n[!] List Empty [!]\n");
    }
    else
    {

        P = L.first;
        while(P != NULL)
        {
            printf("\nNama Dosen : %s",P->D.nama);
            printf("\nNPP : %d\n",P->D.NPP);
            P = P->next;
        }
    }

}
```

Main.c

```c
#include "header.h"
/* run this program using the console pauser or add your own getch, system("pause") or
input loop */


int main(int argc, char *argv[]) {

    List L;
    Data D;

    int menu,NPP;
    string namaDosen;
    createEmpty(&L);
    do{
        system("cls");
        printf("\n====Linked List====");
        printf("\n[1]. Insert First");
        printf("\n[2]. Insert Last");
        printf("\n[3]. Delete First");
        printf("\n[4]. Delete Last");
        printf("\n[5]. Show List");
        printf("\n[0]. Exit");
        printf("\n >> ");scanf("%d",&menu);

        switch(menu){
            case 1:
                printf("\nInput Data");
                printf("\nInput Nama Dosen : ");fflush(stdin);gets(namaDosen);
                printf("\nInput NPP Dosen : ");scanf("%d",&NPP);

                D = makeData(namaDosen,NPP);
                insertFirst(&L,D);
                break;

            case 2:
                printf("\nInput Data");
                printf("\nInput Nama Dosen : ");fflush(stdin);gets(namaDosen);
                printf("\nInput NPP Dosen : ");scanf("%d",&NPP);

                D = makeData(namaDosen,NPP);
                insertLast(&L,D);
                break;

            case 3:
                deleteFirst(&L);
                break;

            case 4:
                deleteLast(&L);
                break;

            case 5:
                showList(L);
                break;

            case 0:
                break;

        }getch();


    }while(menu!=0);
    return 0;
}
```

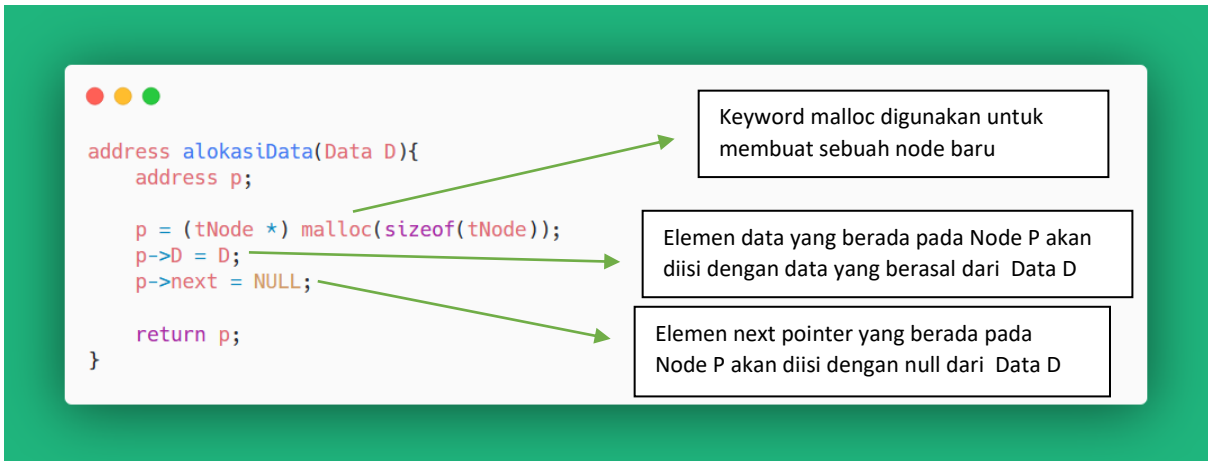
Pembahasan Guided

Pembentukan struktur node

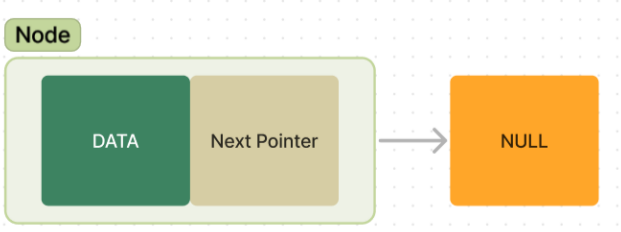


Dapat dilihat pada penjelasan diatas, template code tersebut merupakan pembentukan awal dari sebuah linear linked list.

Alokasi Node Baru

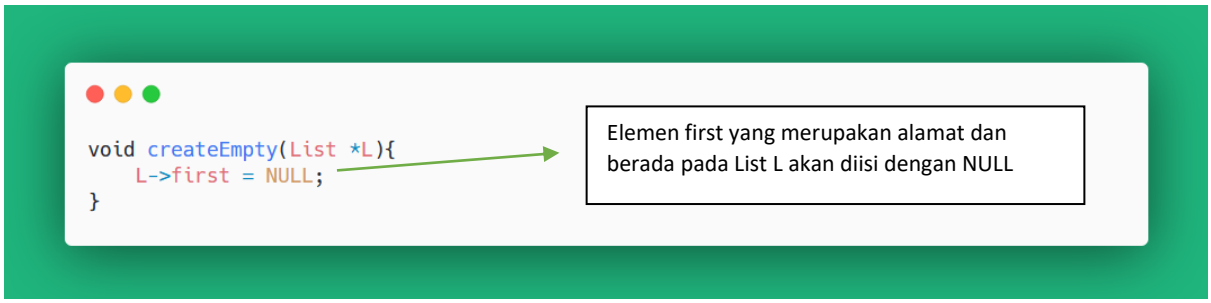


Maka hasil dari fungsi alokasiData akan mereturnkan sebuah node yang berbentuk sebagai berikut:

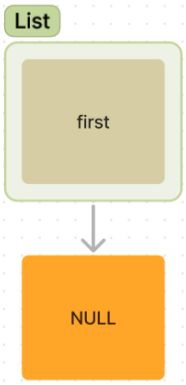


Inisialisasi

Inisialisasi dalam sebuah linked list digunakan untuk mengeset bahwa tidak ada node dalam sebuah linked list, dalam artian lain list tersebut kosong.

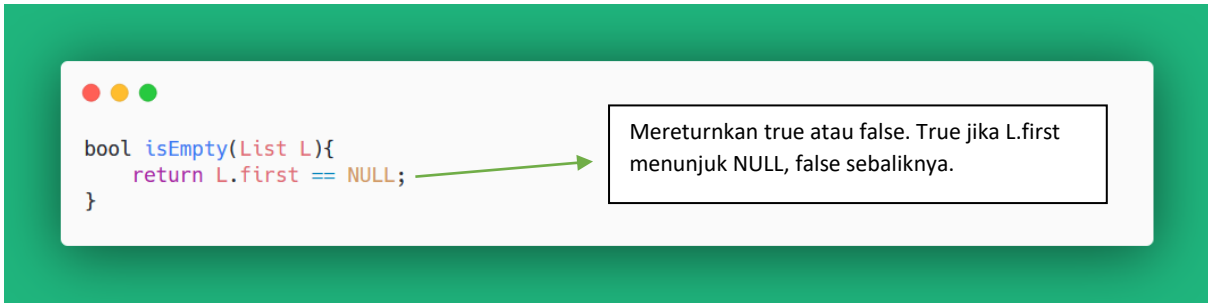


Maka dengan melakukan pemanggilan prosedur createEmpty, akan menghasilkan List sebagai berikut



IsEmpty

IsEmpty digunakan untuk mengecek apakah list tersebut kosong atau tidak, sehingga dapat dilihat jika L.first menunjuk NULL, maka list tersebut kosong, sebaliknya maka list tersebut ada isinya



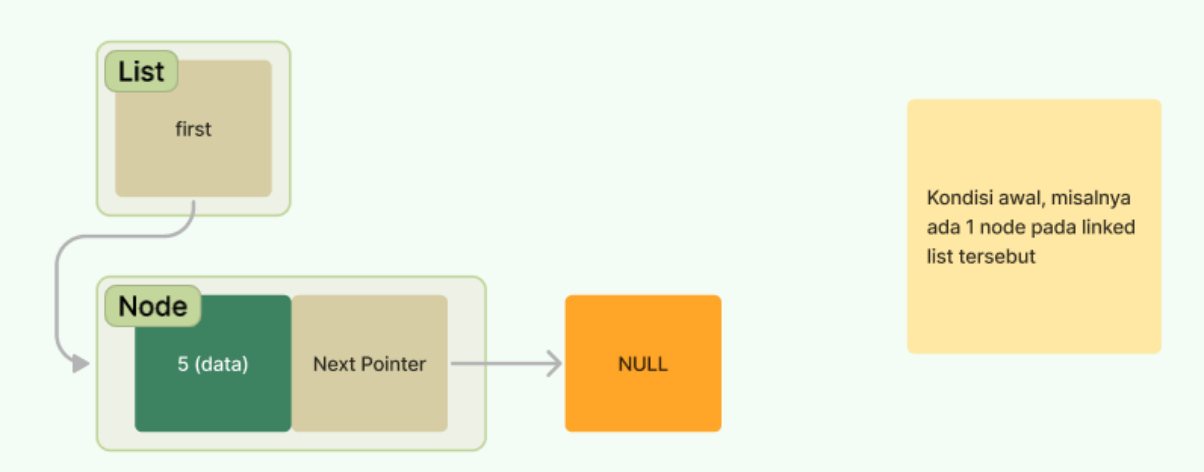
NB : Untuk pembahasan berikutnya, teman teman dapat melihat screenshot gambar dibawah atau dapat juga melihat pada tautan berikut

https://www.figma.com/file/yYz7q8QBVODI565xebVEJM/Linked-List?node-id=0%3A1

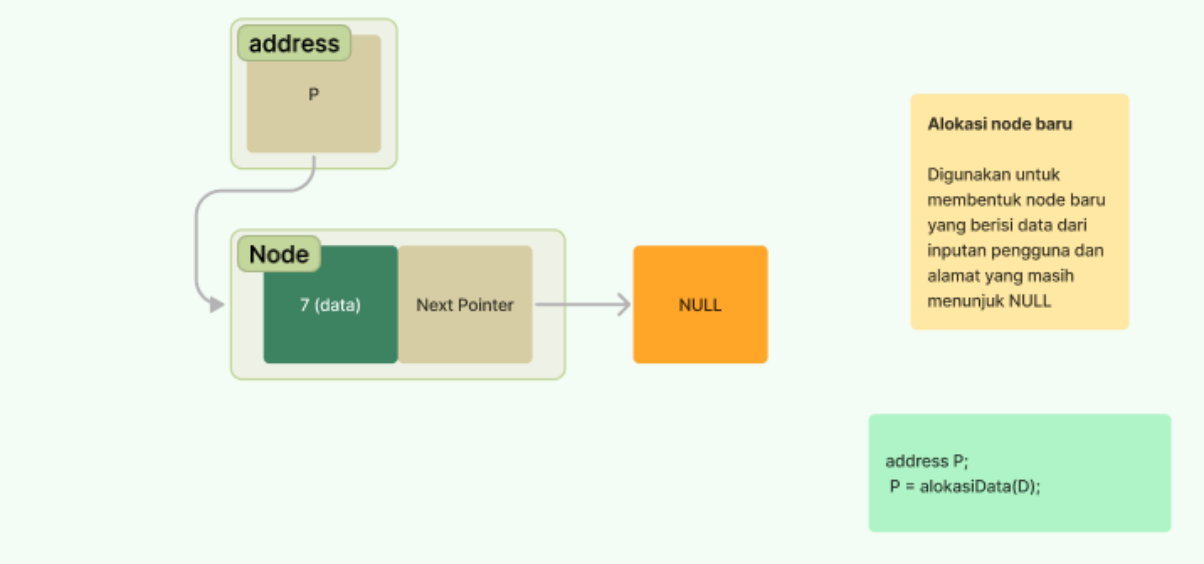
Insert First

Insert First digunakan untuk memasukan data kedalam linked list namun data yang diinputkan akan diletakan paling depan. Berikut rincian langkah langkah nya

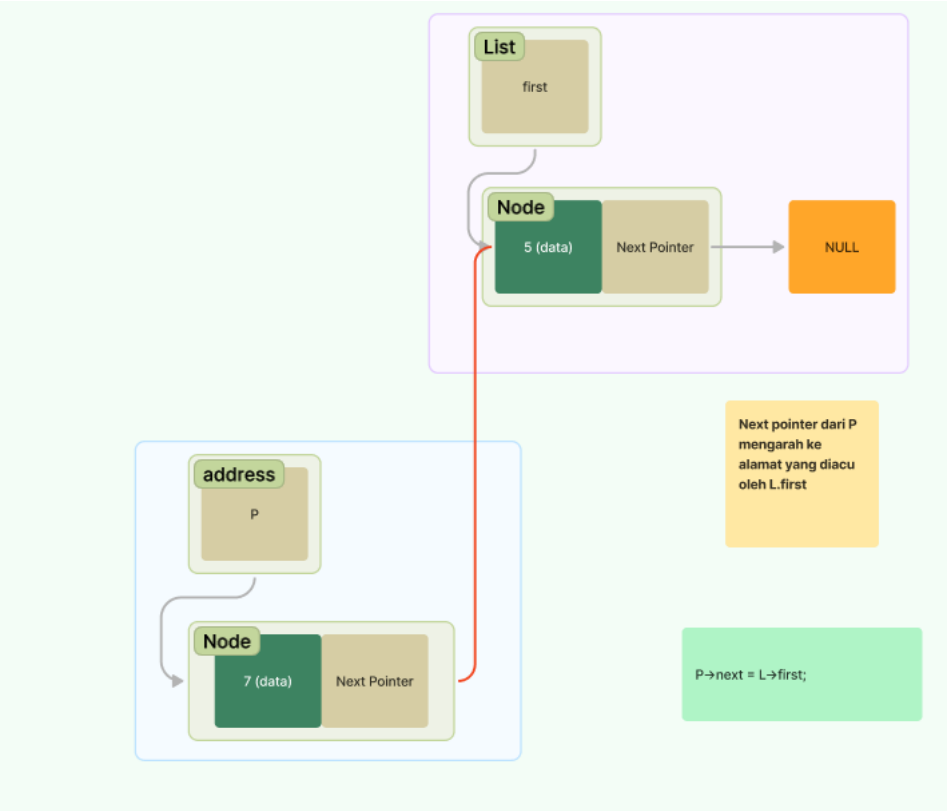
Langkah 1



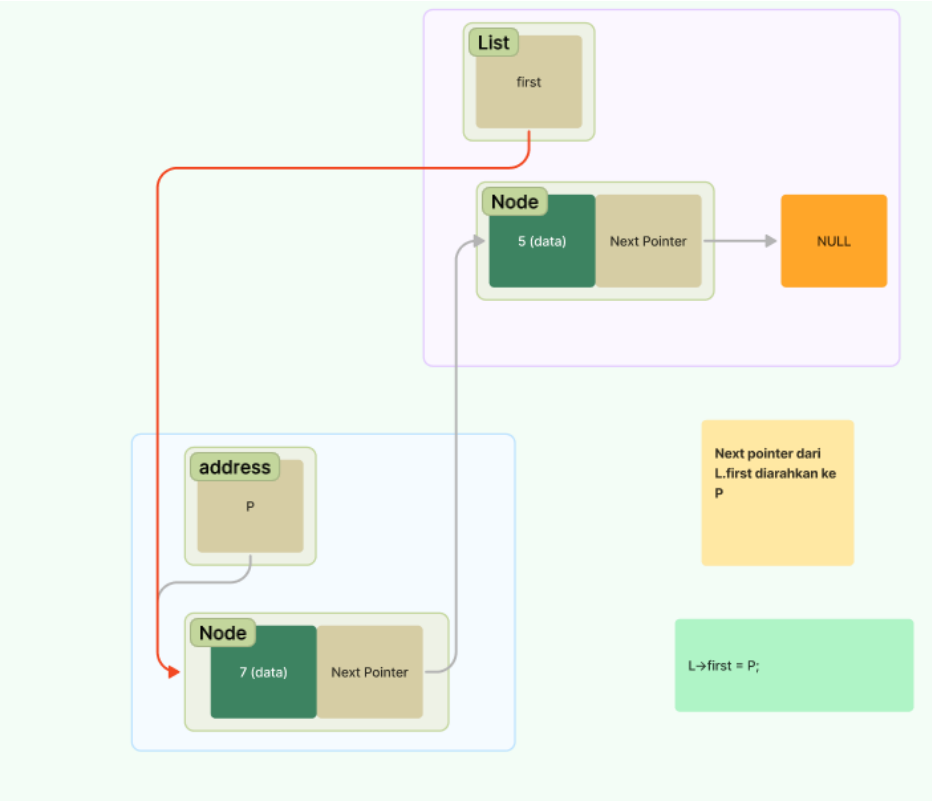
Langkah 2



Langkah 3

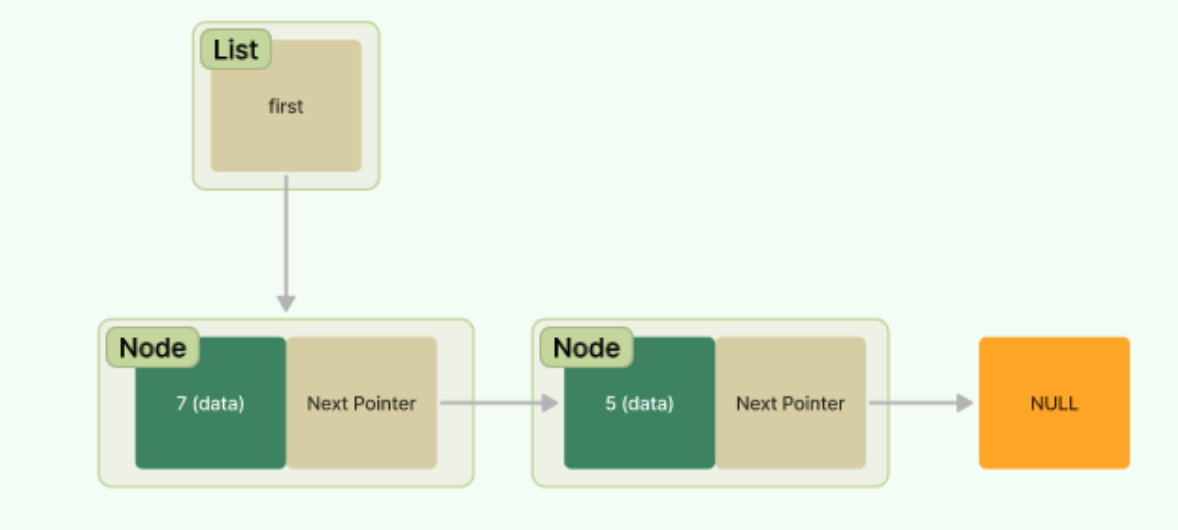


Langkah 4



Langkah 5

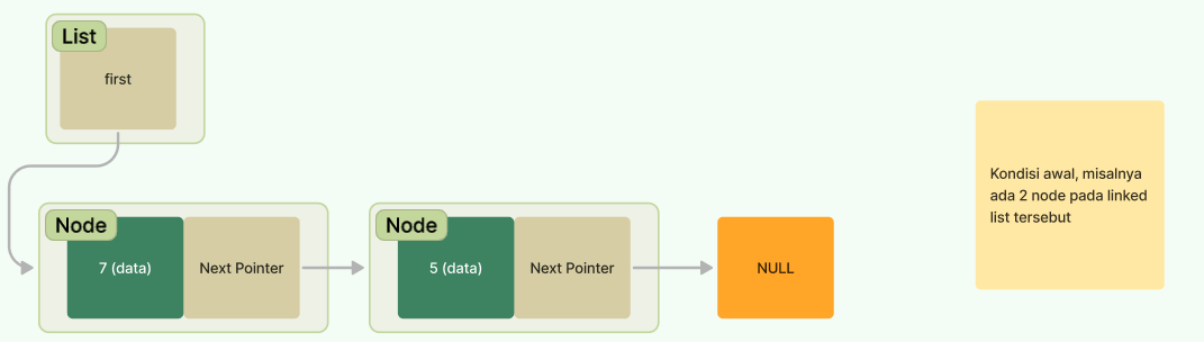
Setelah melakukan insert first, bentuk liinked list akan menjadi seperti berikut



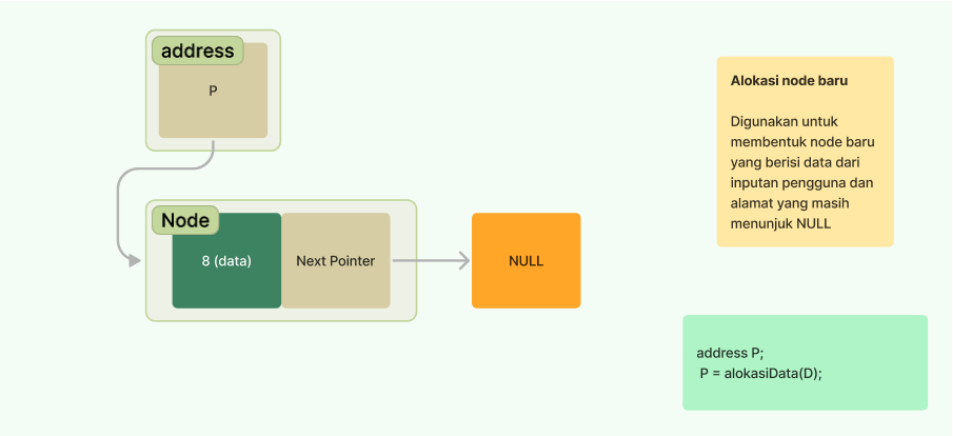
Insert Last

Insert Last digunakan untuk memasukan data kedalam linked list namun data yang diinputkan akan diletakan paling belakang. Berikut rincian langkah langkah nya

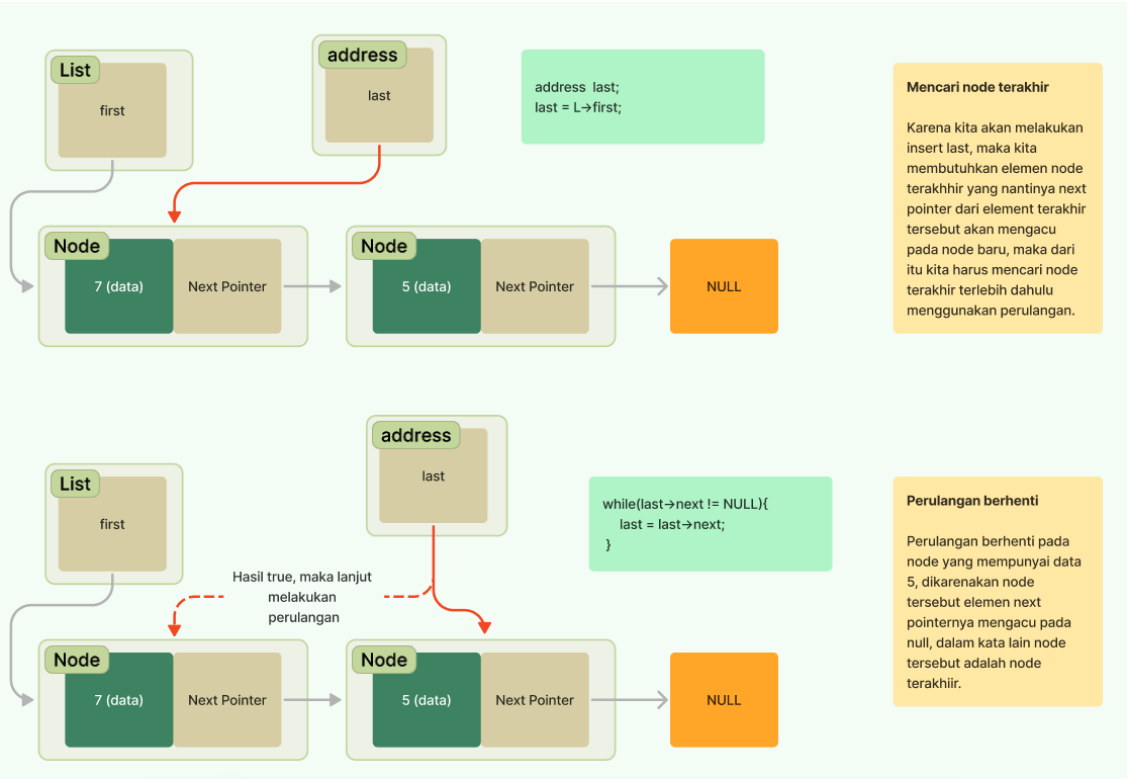
Langkah 1



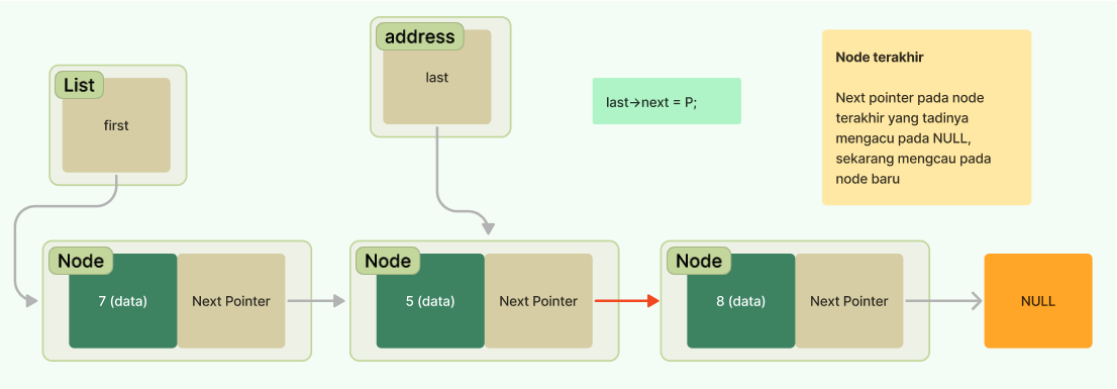
Langkah 2



Langkah 3

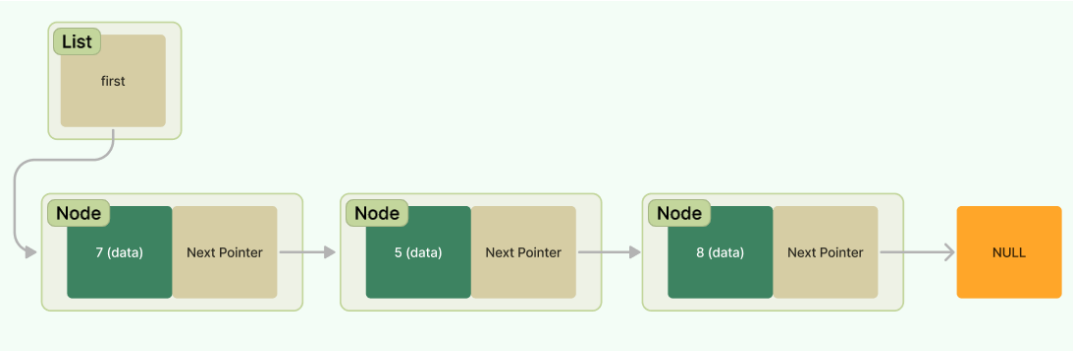


Langkah 4



Langkah 5

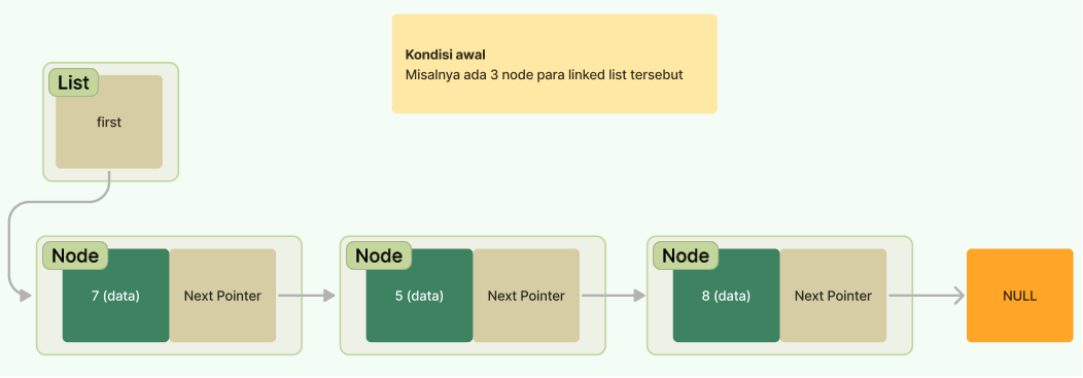
Setelah melakukan insert last, bentuk liinked list akan menjadi seperti berikut



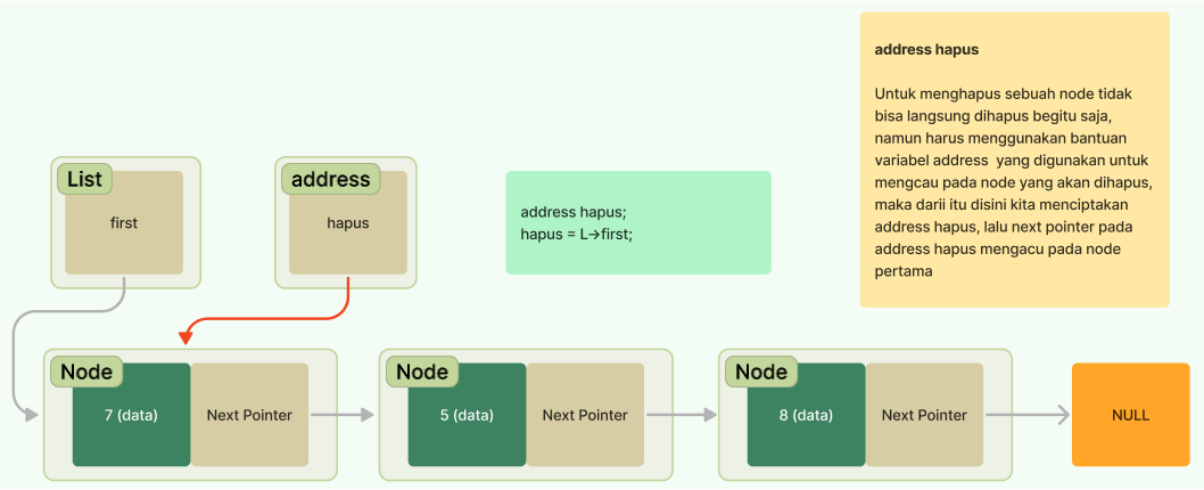
Delete First

Delete first merupakan penghapusan node dalam linked list, namun node yang dihapus adalah node pertama pada linked list tersebut.

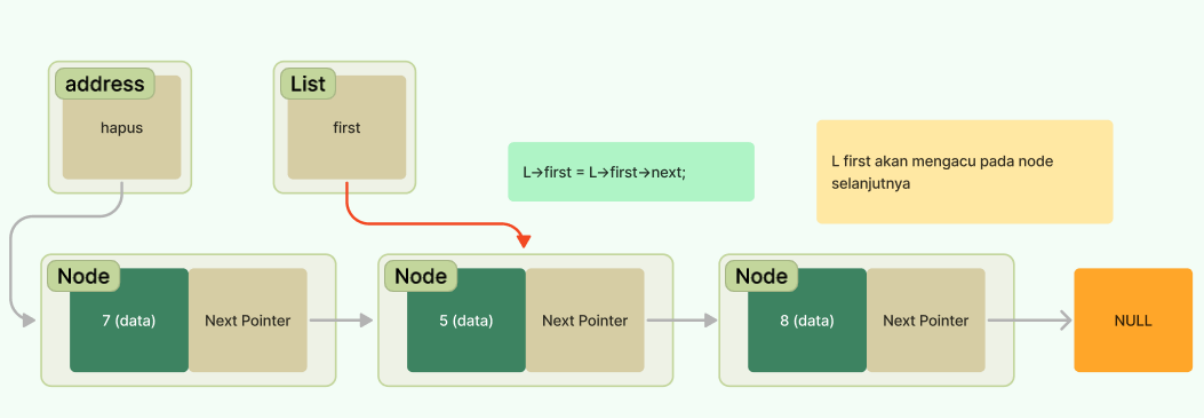
Langkah 1



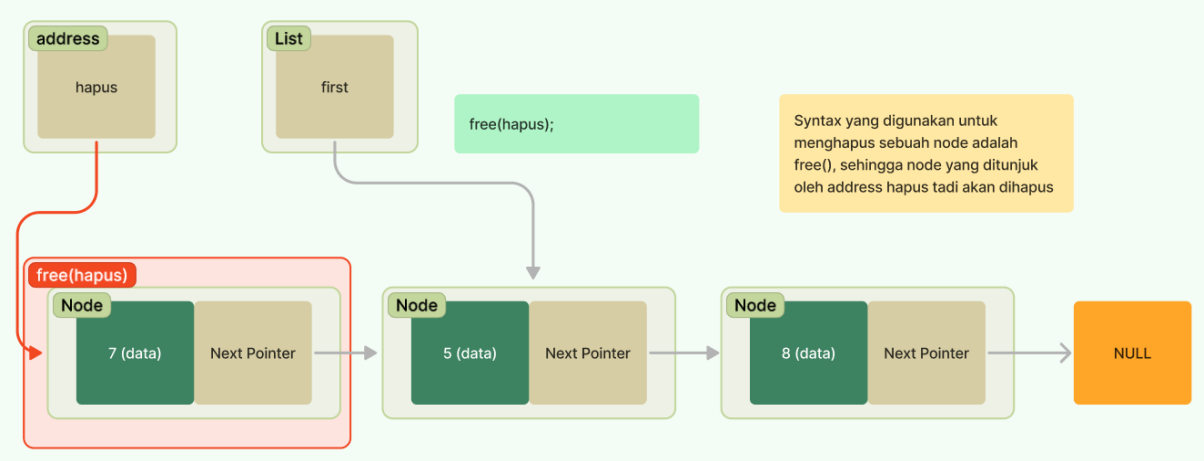
Langkah2



Langkah 3

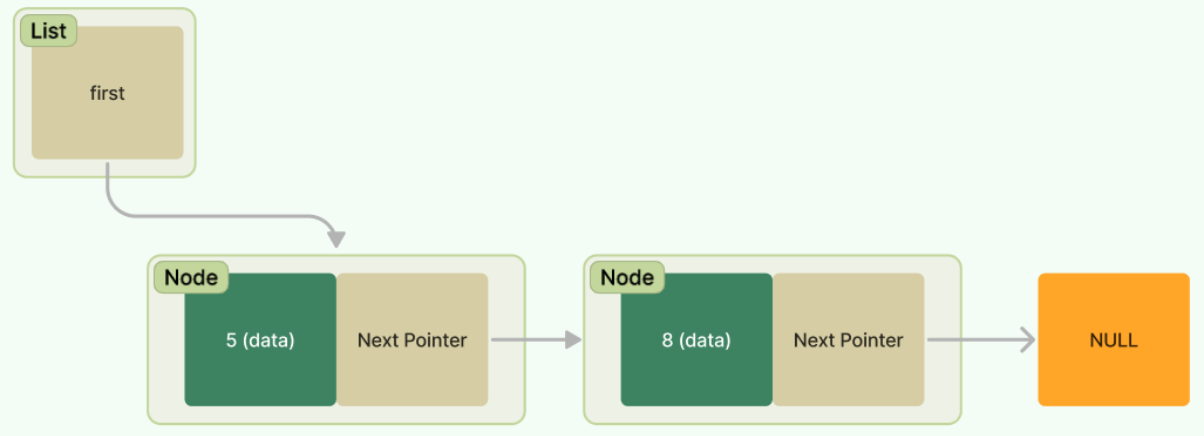


Langkah 4



Langkah 5

Setelah melakukan delete first, kondisi linked list akan menjadi sebagai berikut



Delete Last

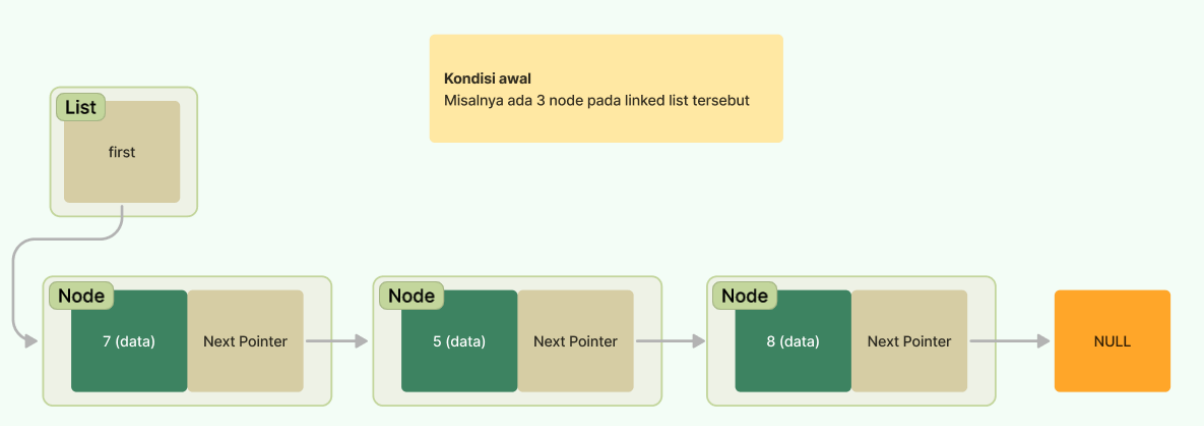
Delete last merupakan penghapusan node dalam linked list, namun node yang dihapus adalah node terakhir pada linked list tersebut.

Beberapa hal yang perlu diperhatikan dalam delete last:

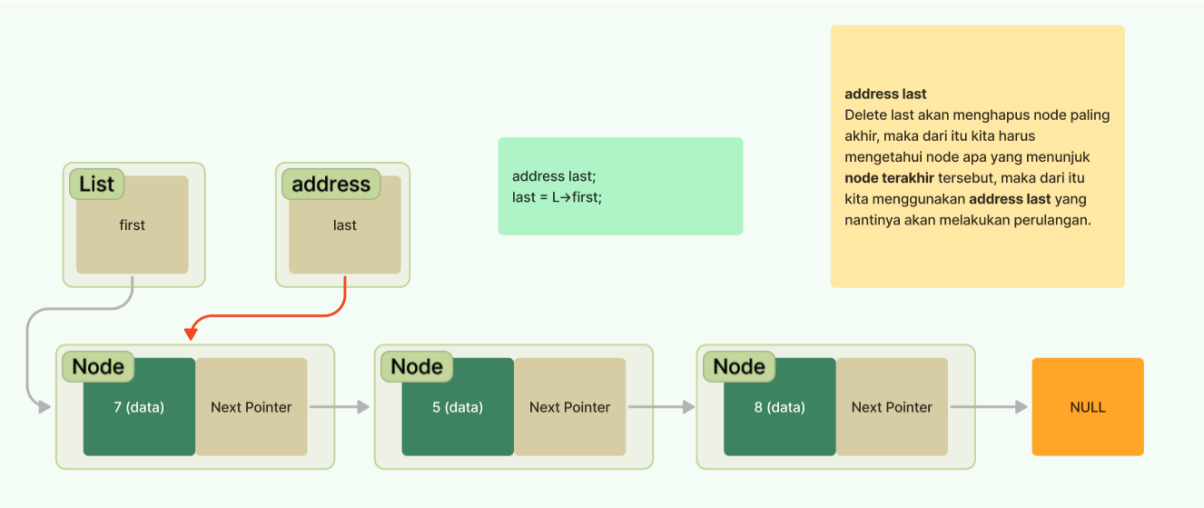
1. Jika list hanya berisi 1 node, maka akan dilakukan delete first
2. Memerlukan sebuah variabel address yang nantinya akan menunjuk node sebelum node terakhir

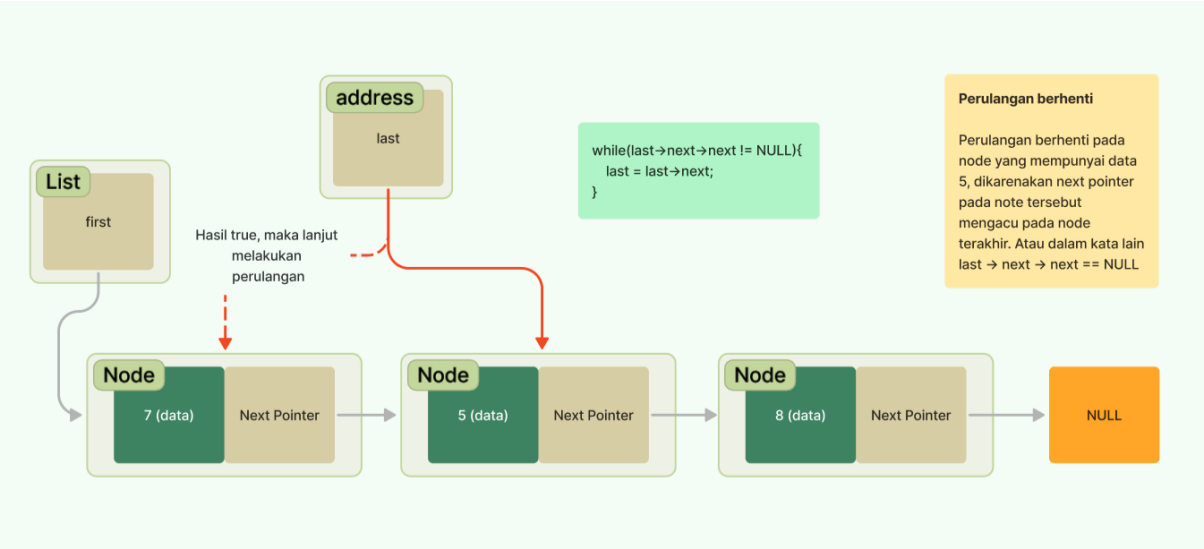
Berikut langkah – langkah nya:

Langkah 1

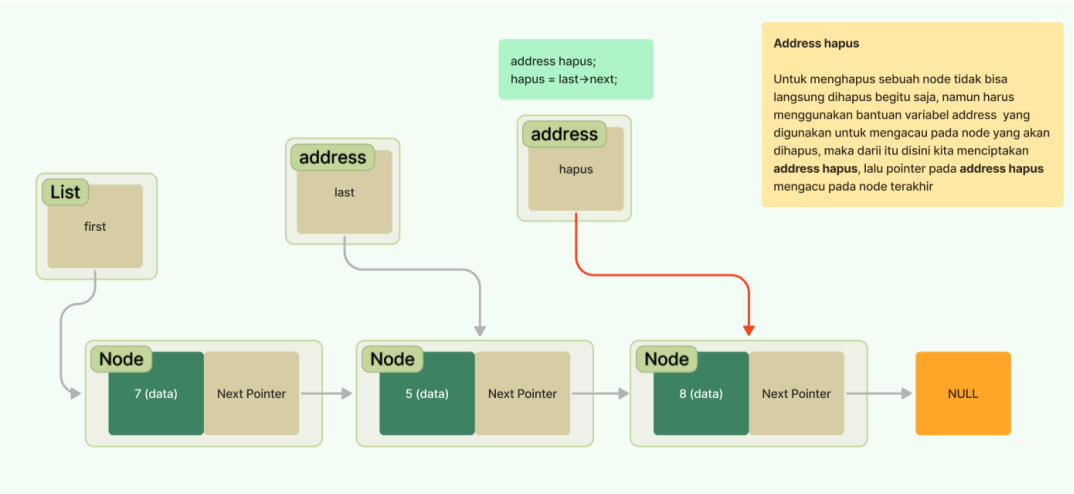


Langkah 2

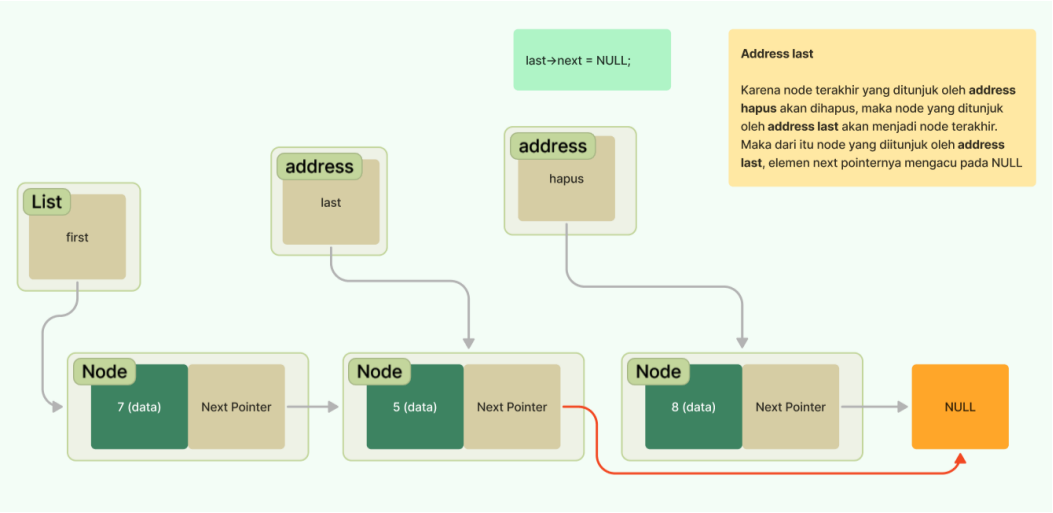




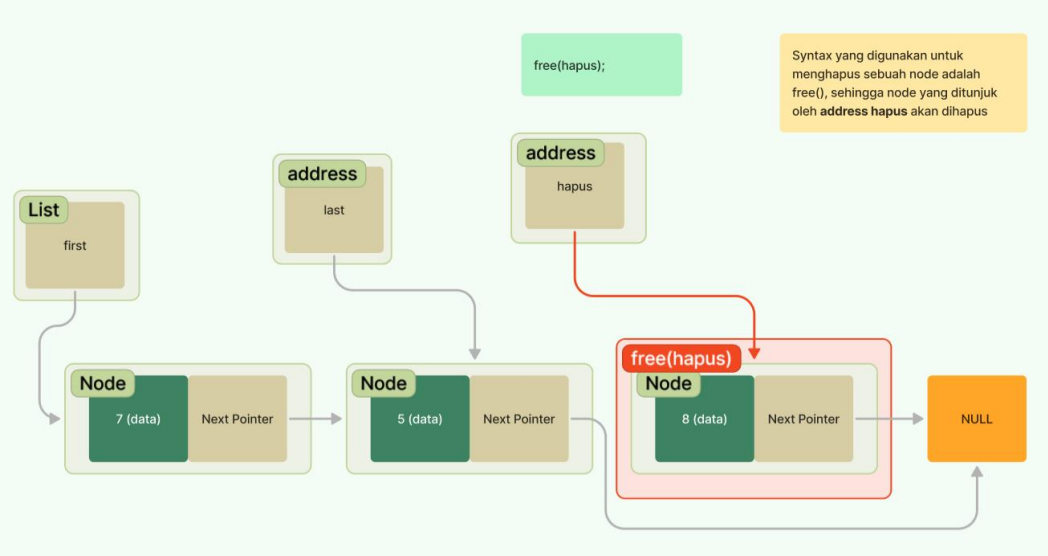
Langkah 3



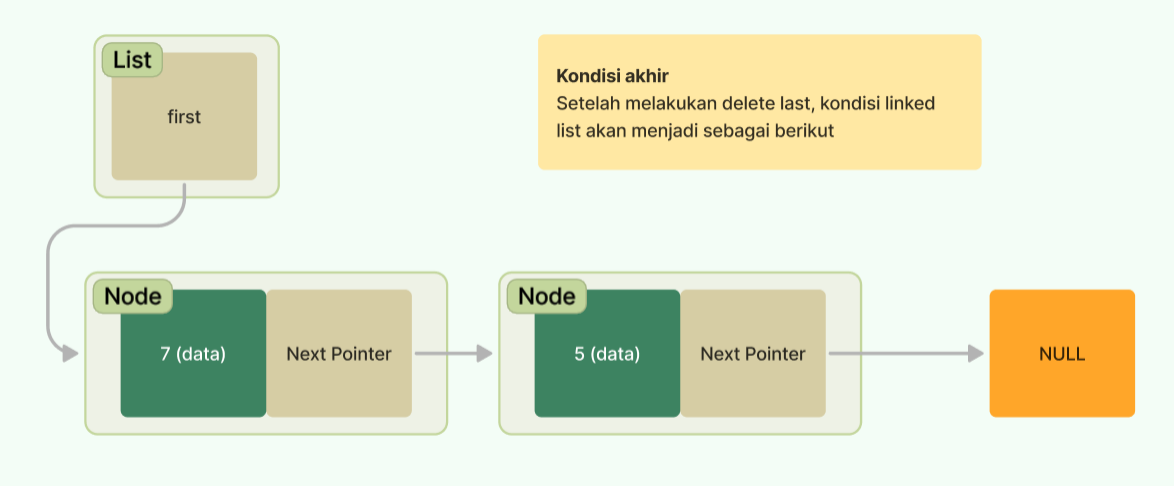
Langkah 4



Langkah 5

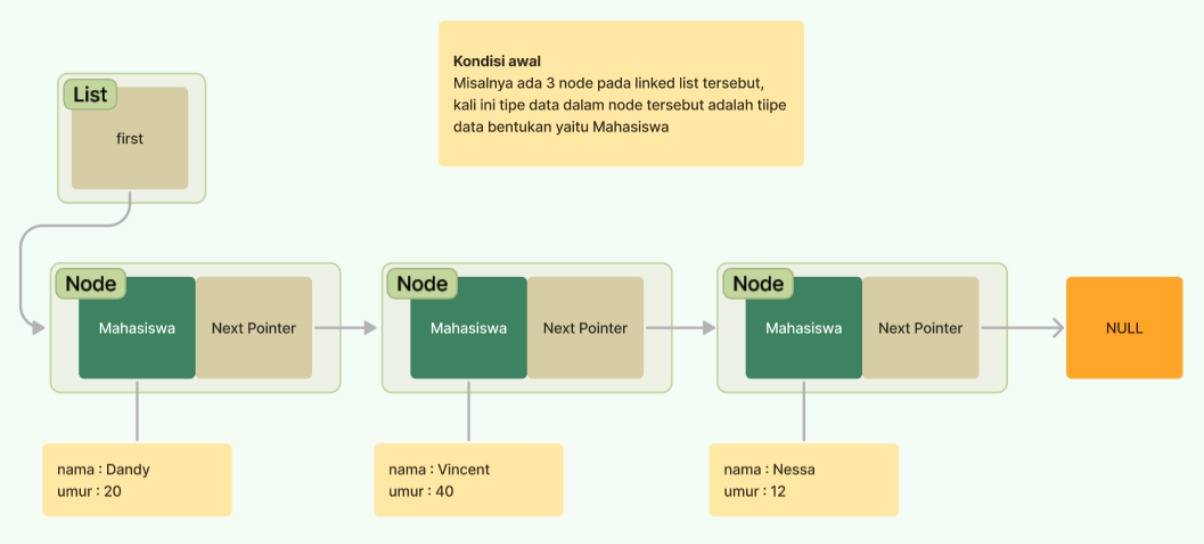


Langkah 6

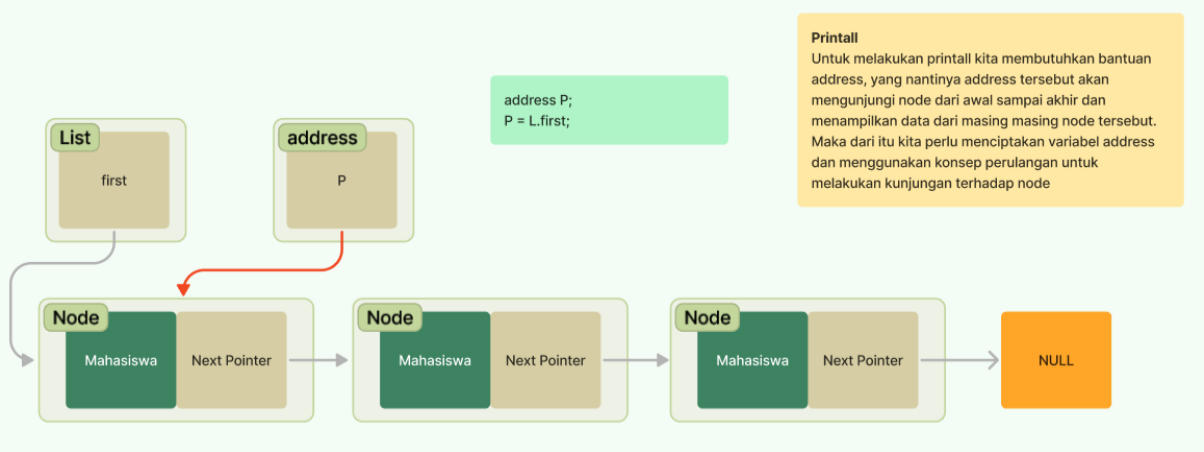


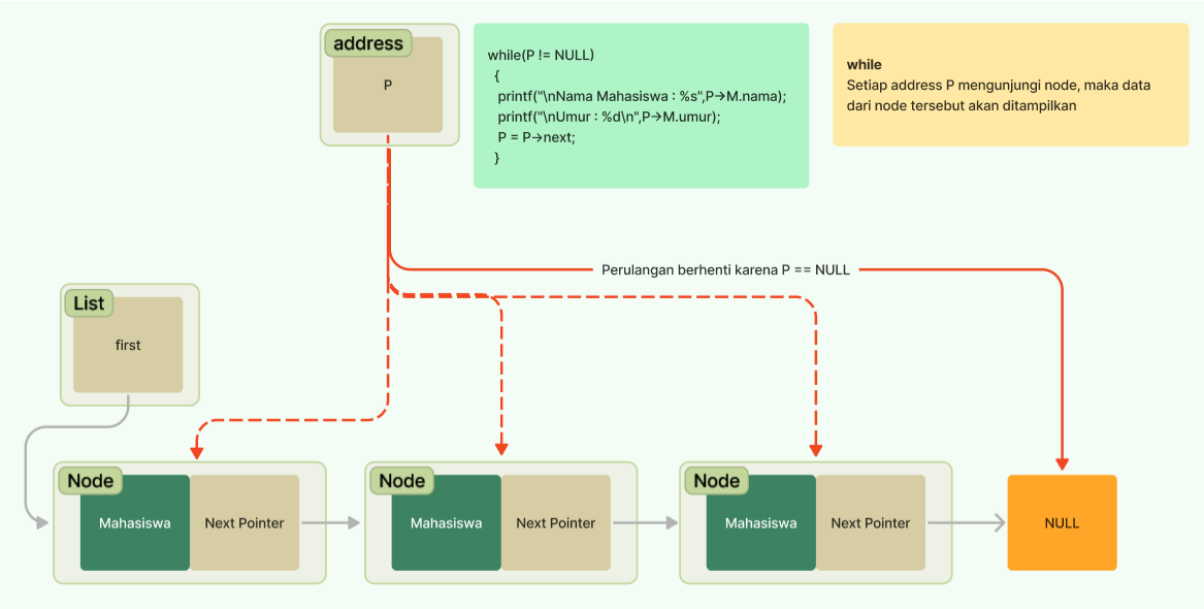
ShowList / PrintAll

Langkah 1



Langkah 2





Pembahasan modul linked list dan guided sudah selesai, jika teman teman masih bingung atau ada yang mau ditanyakan bisa langsung chat ke aku atau asisten yang lain. Terima kasih.