STRUKTUR DATA

LIST GANDA REPRESENTASI STATIS

Rosa Ariani Sukamto

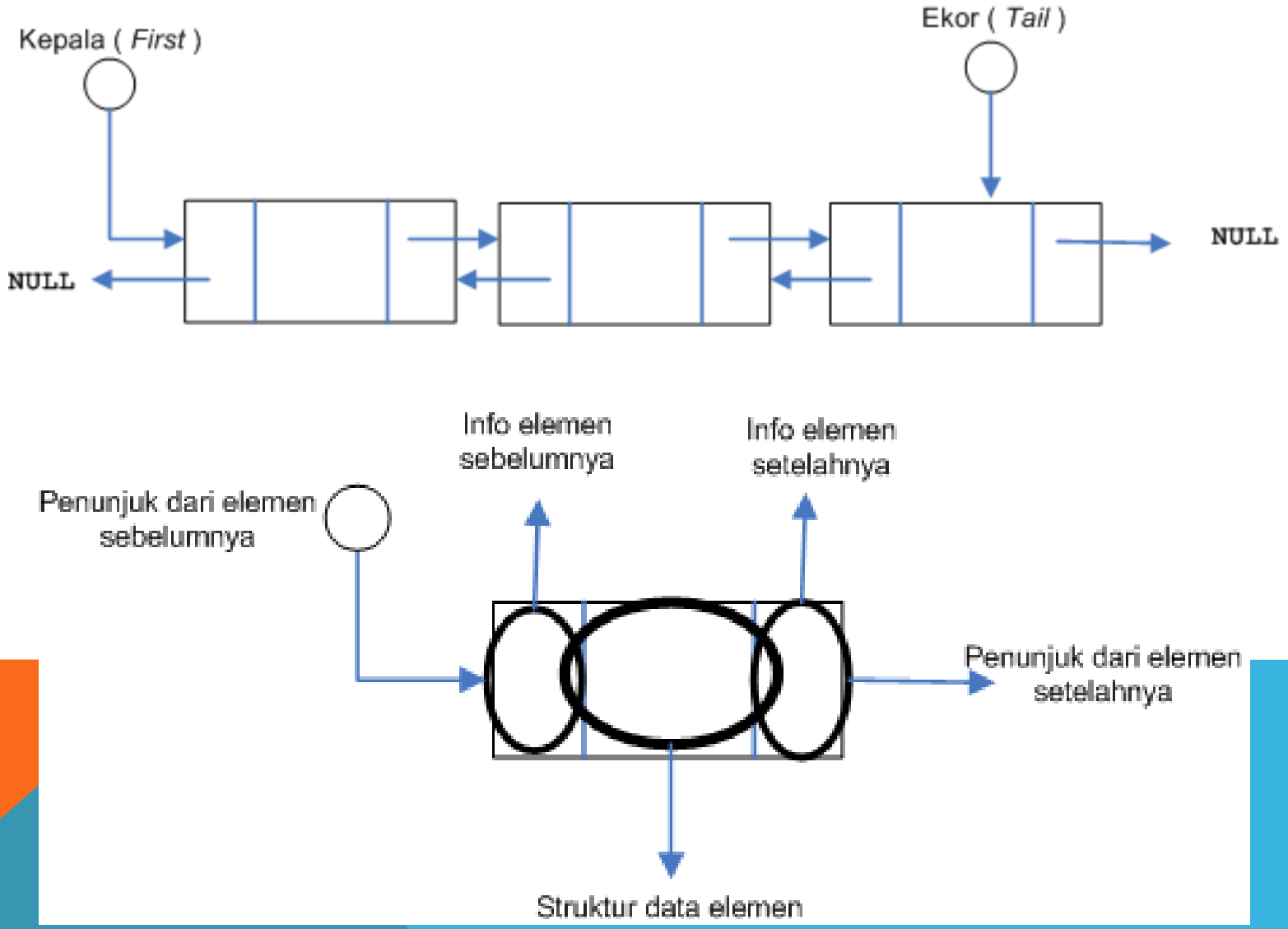
ROSA ARIANI SUKAMTO

Blog: http://hariiniadalahhadiah.wordpress.com

Facebook: https://www.facebook.com/rosa.ariani.sukamto

Email: rosa_if_itb_01@yahoo.com

LIST GANDA


Kepala (First)
Ekor (Tail)
NULL
NULL
Info elemen sebelumnya
Info elemen setelahnya
Penunjuk dari elemen sebelumnya
Penunjuk dari elemen setelahnya
Struktur data elemen

DEKLARASI ELEMEN

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

typedef struct{
  char nim[10];
  char nama[50];
  char nilai[2];
}nilaiMatKul;
```

```
typedef struct{
  nilaiMatKul elmt;
  int prev;
  int next;
}elemen;

typedef struct{
  int first;
  int tail;
  elemen data[10];
}list;
```

CREATE LIST

```
void createList(list *L){

   (*L).first = -1;
   (*L).tail = -1;
   int i;

   for(i=0;i<10;i++){
      /*proses menginisialisasi isi array*/
      (*L).data[i].prev = -2;
      (*L).data[i].next = -2;
   }
}
```

COUNT ELEMENT

```
int countElement(list L){
  int hasil = 0;
  if(L.first != -1){
    /*list tidak kosong*/
    int elmt;

    /*inisialisasi*/
    elmt = L.first;

    while(elmt != -1){
      /*proses*/
      hasil = hasil + 1;

      /*iterasi*/
      elmt = L.data[elmt].next;
    }
  }
```

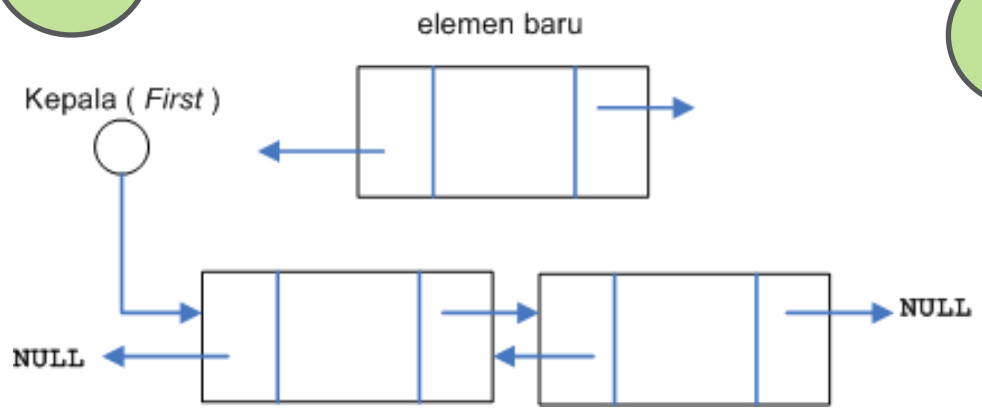
```
    return hasil;
}
```

EMPTY ELEMENT

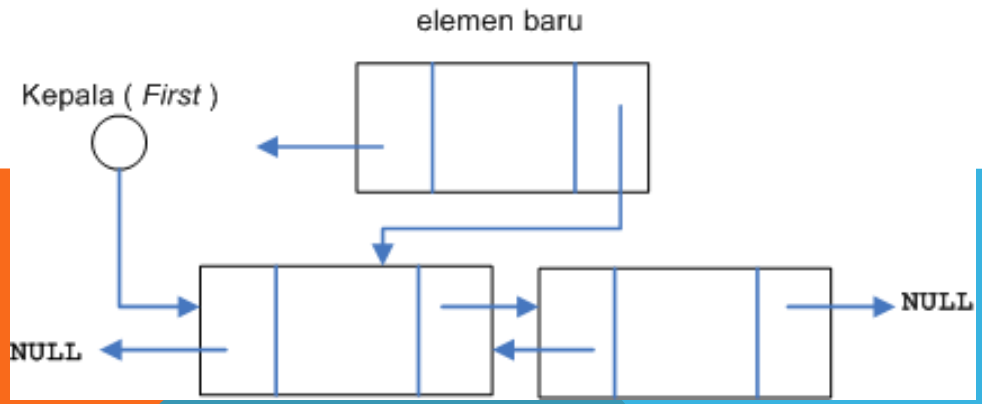
```
int emptyElement(list L){
  int hasil = -1;

  if(countElement(L) < 10){
    int ketemu = 0;

    int i = 0;
    while((ketemu == 0)&&(i <
 10)){
      if(L.data[i].next == -2){
        hasil = i;
        ketemu = 1;
      }
      else{
        i = i + 1;
      }
    }
  }
    return hasil;
}
```

PENAMBAHAN ELEMEN DI AWAL LIST (ADDFIRST)

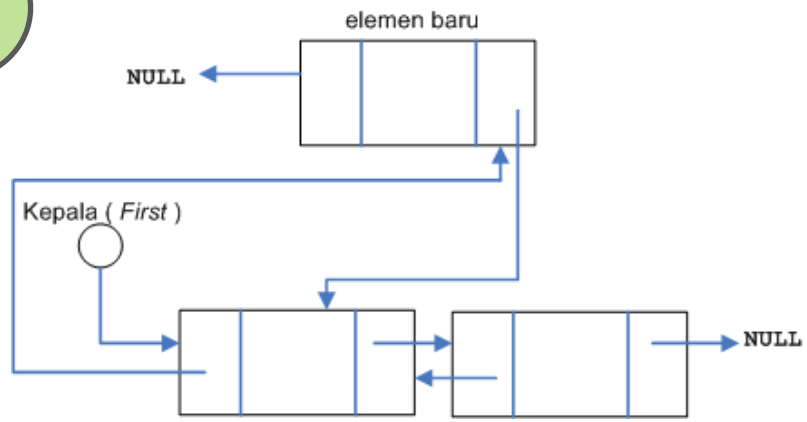
1



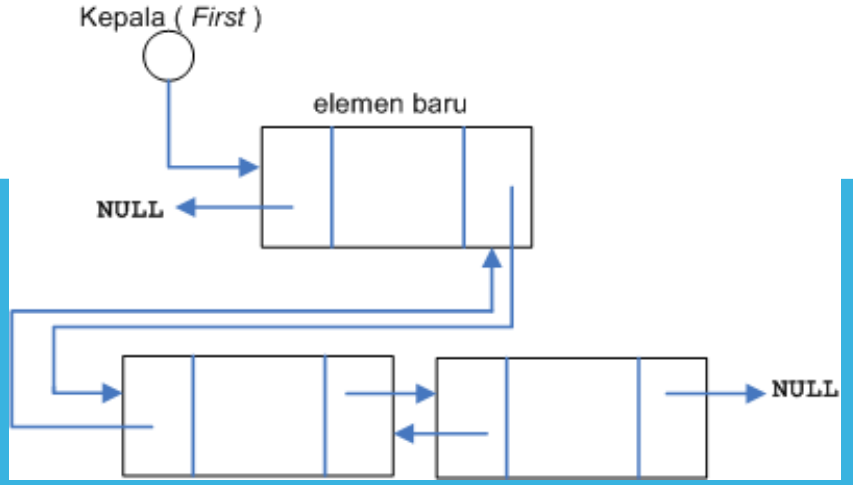
2



3



4



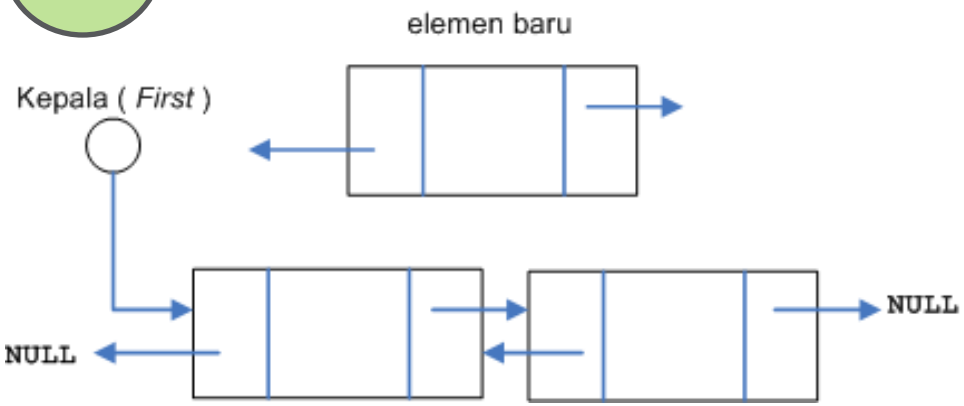
ADD FIRST

```
void addFirst(char nim[], char
   nama[], char nilai[], list *L){
  if(countElement(*L) < 10){
    int baru = emptyElement(*L);

   strcpy((*L).data[baru].elmt.nim,
   nim);
   strcpy((*L).data[baru].elmt.nama,
   nama);
   strcpy((*L).data[baru].elmt.nilai
   , nilai);

    if((*L).first == -1){
      /*jika list kosong*/
      (*L).data[baru].prev = -1;
      (*L).data[baru].next = -1;
      (*L).tail = baru;
    }
```

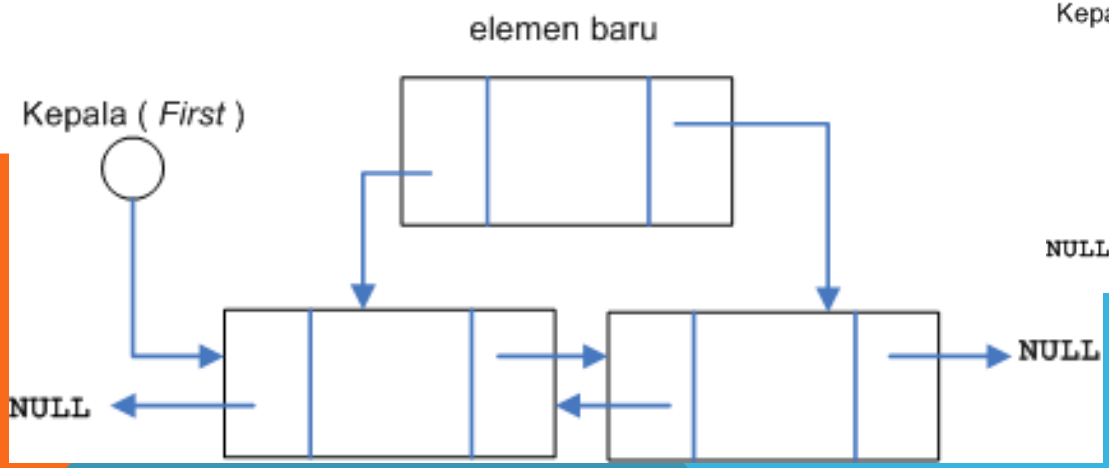
```
  else{
       /*jika list tidak
   kosong*/
    (*L).data[baru].prev = -1;
    (*L).data[baru].next =
    (*L).first;
    (*L).data[(*L).first].prev
    = baru;
    }

    (*L).first = baru;
  }
  else{
    /*proses jika array
   penuh*/
    printf("sudah tidak dapat
   ditambah\n");
  }
}
```

PENAMBAHAN ELEMEN DI TENGAH (ADDAFTER)

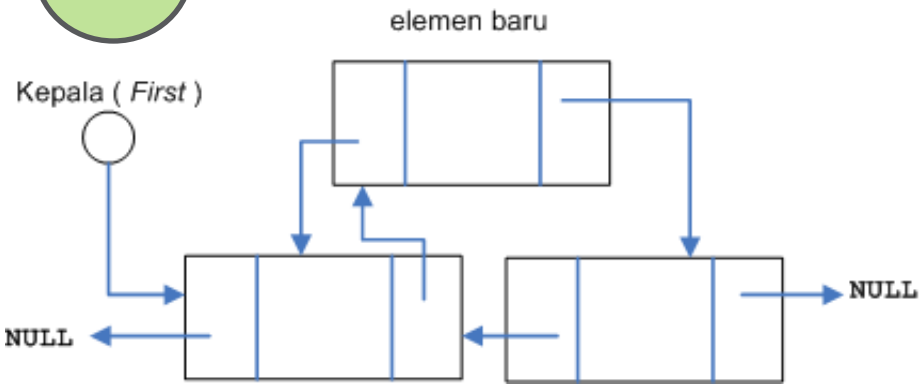
1



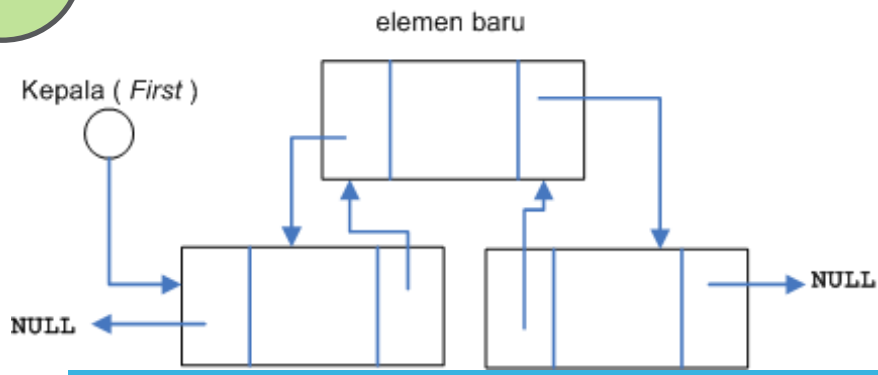
2



3



4

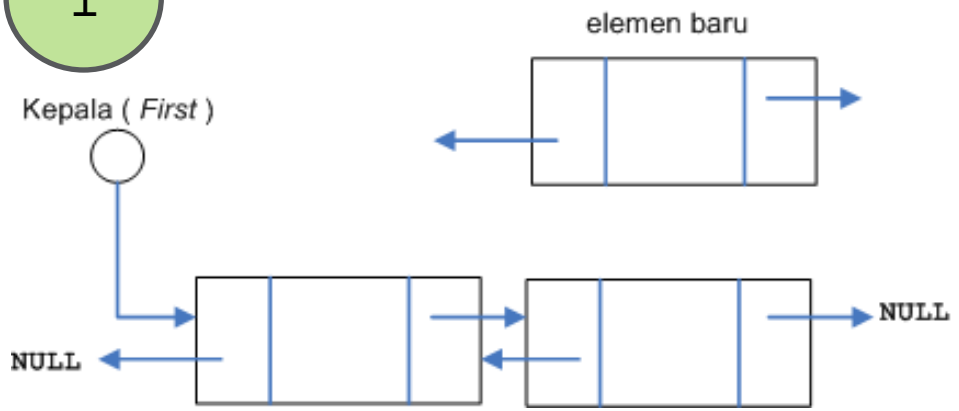


ADD AFTER

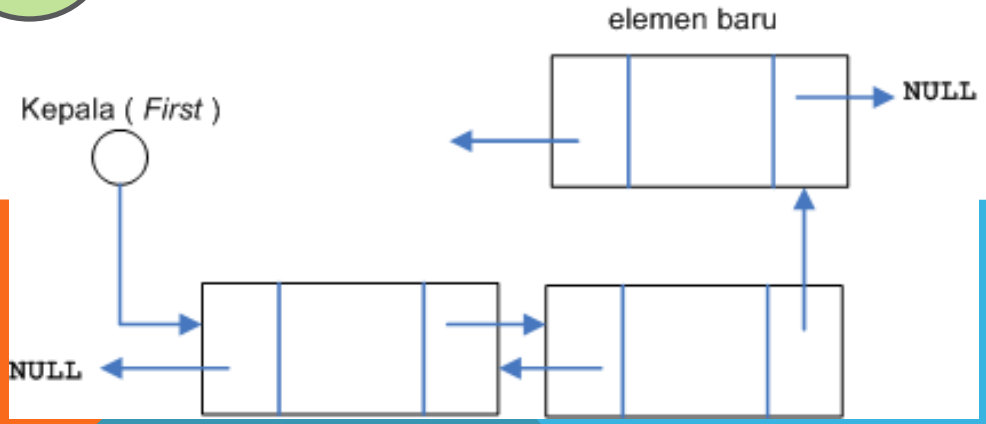
```
void addAfter(int prev, char
   nim[], char nama[], char
   nilai[], list *L){


  if(countElement(*L) < 10){

    int baru = emptyElement(*L);


strcpy((*L).data[baru].elmt.nim,
   nim);

strcpy((*L).data[baru].elmt.nama
   , nama);

strcpy((*L).data[baru].elmt.nila
   i, nilai);

if((*L).data[prev].next != -1){

//jika baru bukan menjadi elemen
   terakhir

   (*L).data[baru].prev = prev;

   (*L).data[baru].next =
   (*L).data[prev].next;

   (*L).data[prev].next = baru;
   (*L).data[(*L).data[baru].next].pr
   ev = baru;

   }else{

   //jika baru menjadi elemen
   terakhir

     (*L).data[baru].prev = prev;

     (*L).data[prev].next = baru;

     (*L).data[baru].next = -1;

     (*L).tail = baru;

   }

}else{

     /*proses jika array penuh*/

     printf("sudah tidak dapat
   ditambah\n");

  }

}
```

PENAMBAHAN ELEMEN DI AKHIR (ADDLAST)

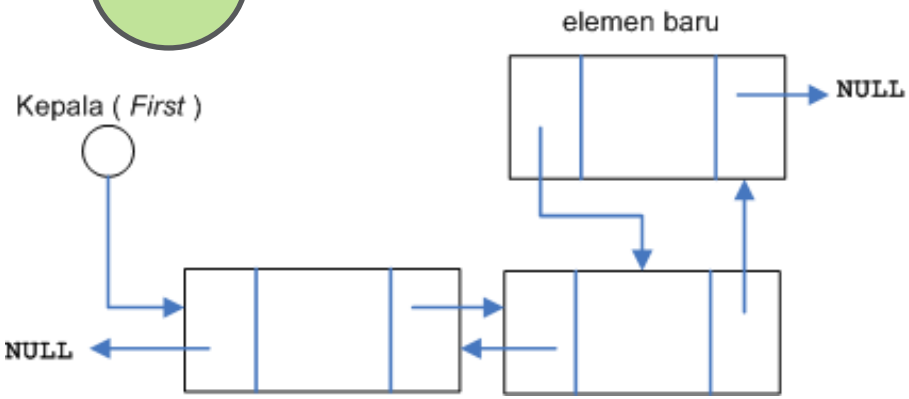
1



2



3



ADD LAST (1)

```
void addLast(char nim[], char nama[],
    char nilai[], list *L){
  if((*L).first == -1){
    /*proses jika list masih kosong*/

    int baru = 0;
strcpy((*L).data[baru].elmt.nim,
    nim);
strcpy((*L).data[baru].elmt.nama,
    nama);
strcpy((*L).data[baru].elmt.nilai,
    nilai);

(*L).data[baru].prev = -1;
(*L).data[baru].next = -1;
(*L).first =  baru;
(*L).tail =  baru;

}
```

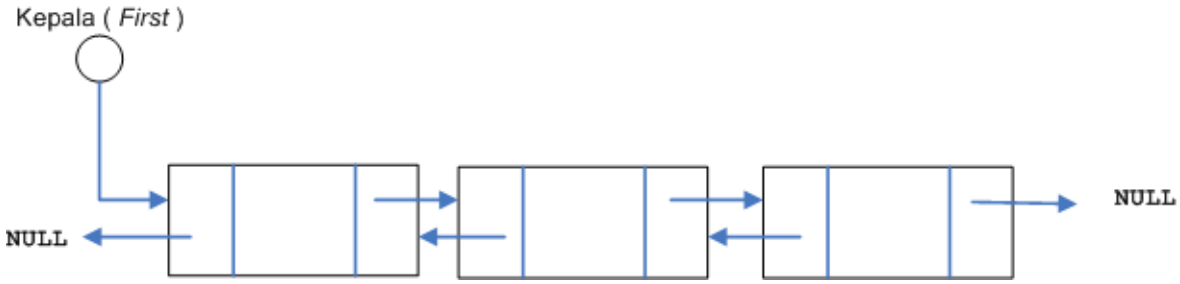
```
else{
  /*proses jika list telah berisi
   elemen*/
    if(countElement(*L) < 10){
      /*proses jika array belum
    penuh*/
      int baru =
    emptyElement(*L);

strcpy((*L).data[baru].elmt.nim,
    nim);
strcpy((*L).data[baru].elmt.nama,
    nama);
strcpy((*L).data[baru].elmt.nilai
    , nilai);
(*L).data[baru].next = -1;
```

ADD LAST (2)

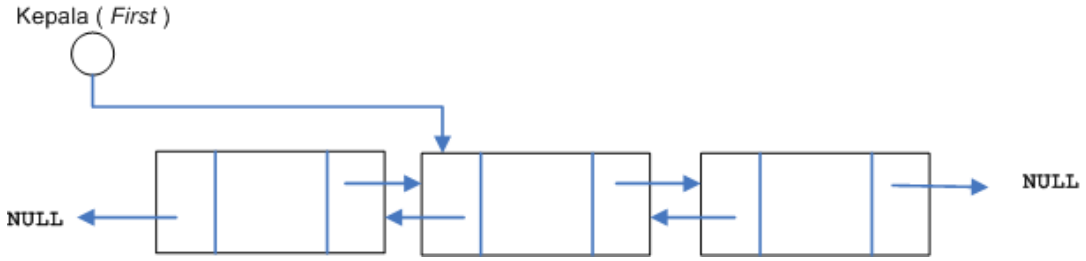
```
        (*L).data[(*L).tail].next = baru;
        (*L).data[baru].prev = (*L).tail;
        (*L).tail = baru;
}
 else{
        /*proses jika array penuh*/
        printf("sudah tidak dapat ditambah\n");
     }
   }
}
```

HAPUS ELEMEN AWAL (DELFIRST)

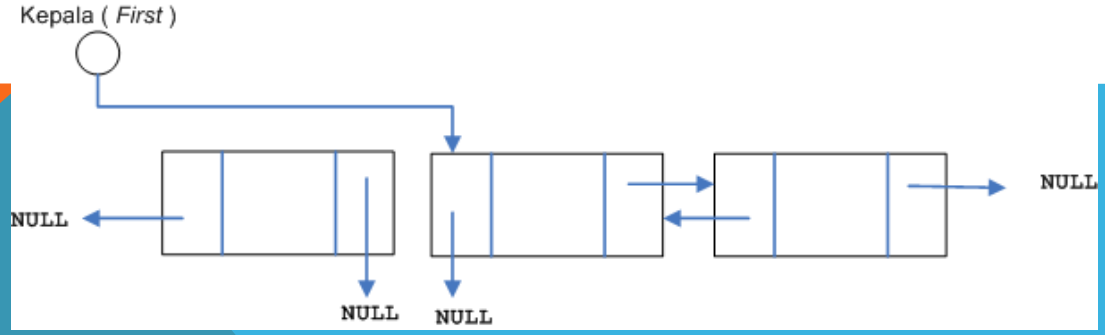
1



2



3

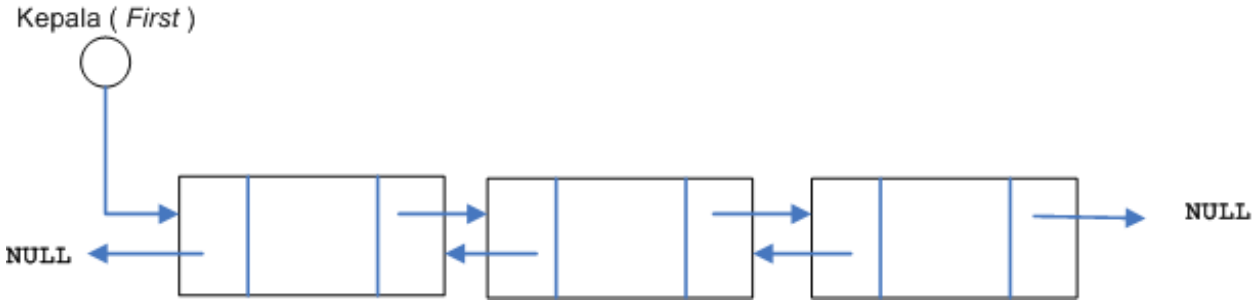


DEL FIRST

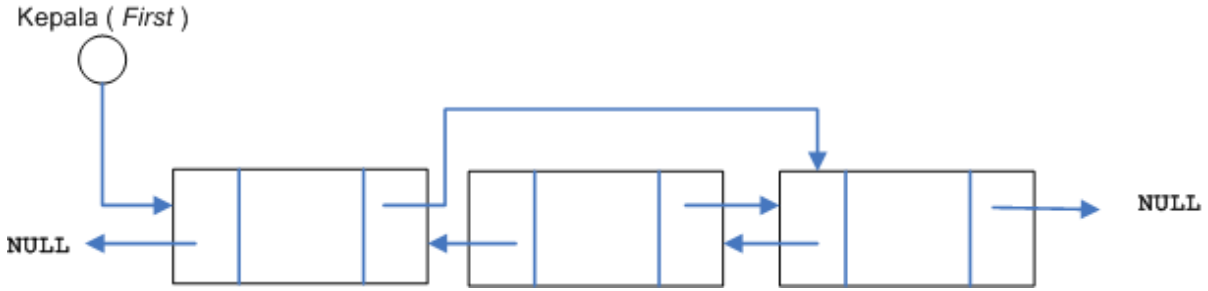
```
void delFirst(list *L){


  if((*L).first != -1){
    int hapus = (*L).first;
    if(countElement(*L) == 1){
        (*L).first = -1;
        (*L).tail = -1;
    }else{
        (*L).first = (*L).data[(*L).first].next;
        (*L).data[(*L).first].prev = -1;
    }
    /*elemen awal sebelumnya dikosongkan*/
    (*L).data[hapus].prev = -2;
    (*L).data[hapus].next = -2;
  }
  else{
   /*proses jika list kosong*/
    printf("list kosong\n");
  }
}
```

HAPUS ELEMEN TENGAH (DELAFTER)

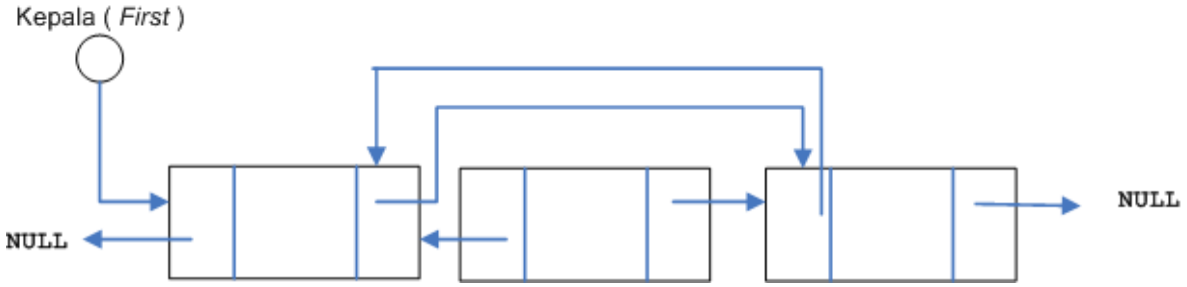
1



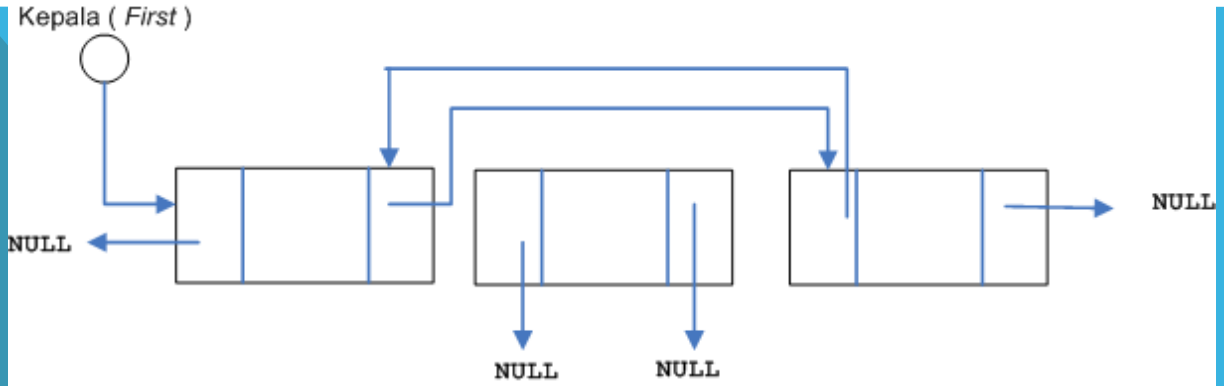
2



3



4

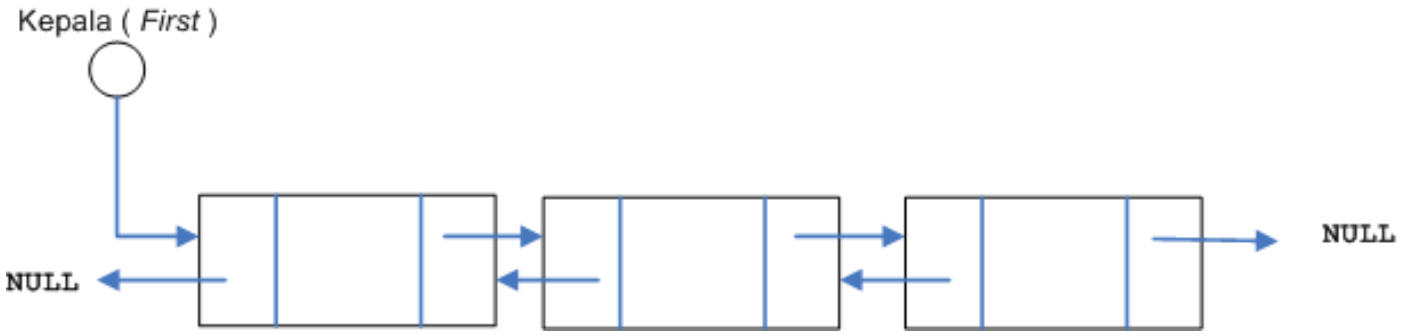


DEL AFTER

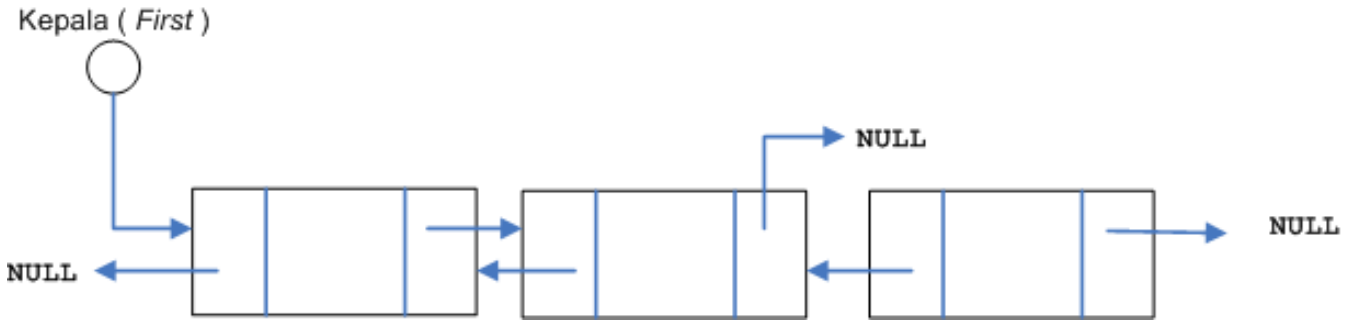
```
void delAfter(int prev, list *L){

  int hapus = (*L).data[prev].next;
  if(hapus != -1){
    if((*L).data[hapus].next == -1){
       (*L).tail = prev;
       (*L).data[prev].next = -1;
    }else{
      (*L).data[prev].next = (*L).data[hapus].next;
      (*L).data[(*L).data[hapus].next].prev = prev;
    }

    /*pengosongan elemen*/
    (*L).data[hapus].prev = -2;
    (*L).data[hapus].next = -2;
  }

}
```

HAPUS DI AKHIR (DELLAST)

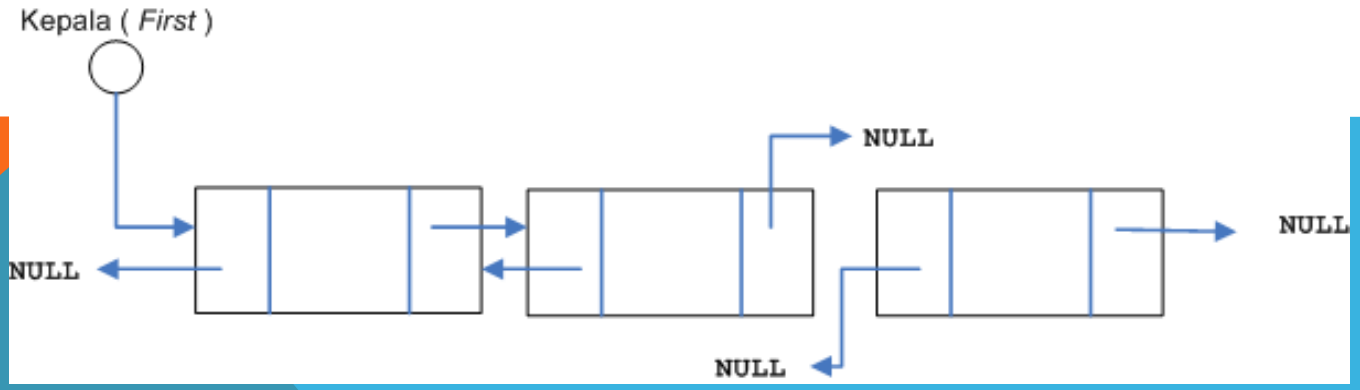
1



2



3



DEL LAST

```
void delLast(list *L){
 if((*L).first != -1){
   if(countElement(*L) == 1){
     /*proses jika list hanya
   berisi satu elemen*/
     delFirst(L);
   }
   else{
    int hapus = (*L).tail;
    (*L).tail =
   (*L).data[hapus].prev;
    (*L).data[(*L).tail].next =
   -1;
     /*elemen terakhir
    sebelumnya dikosongkan*/
   (*L).data[hapus].prev = -2;
   (*L).data[hapus].next = -2;
    }
 }
 else{
  /*proses jika list kosong*/
  printf("list kosong\n");
 }
}
```

PRINT ELEMENT

```
void printElement(list L){
  if(L.first != -1){
    /*inisialisasi*/
    int elmt = L.first;
    int i = 1;
    while(elmt != -1){
      /*proses*/
      printf("elemen ke : %d\n",
   i);
      printf("nim : %s\n",
   L.data[elmt].elmt.nim);
      printf("nama : %s\n",
   L.data[elmt].elmt.nama);
      printf("nilai : %s\n",
   L.data[elmt].elmt.nilai);
      printf("next : %d\n",
   L.data[elmt].next);
      printf("------------
   \n");

      /*iterasi*/
    elmt = L.data[elmt].next;
      i = i + 1;
    }
  }
  else{
   /*proses jika list
   kosong*/
    printf("list kosong\n");
  }
}
```

DEL ALL

```
void delAll(list *L){
  int i;
  for(i=countElement(*L);i>=1;i--){
  /*proses menghapus elemen list*/
    delLast(L);
  }
}
```

MAIN

```
int main(){
  list L;
  createList(&L);
  printElement(L);
  printf("================\n");

  addFirst("1", "Orang_1", "A",
   &L);
  addAfter(L.first, "2",
   "Orang_2", "A", &L);
  addLast("3", "Orang_3", "A",
   &L);
  printElement(L);

  printf("================\n");
  delLast(&L);
  delAfter(L.first, &L);
  delFirst(&L);
  printElement(L);

   printf("================\
   n");

  return 0;

}
```

DAFTAR PUSTAKA

S, Rosa A. dan M. Shalahuddin. 2010. Modul Pembelajaran: Struktur Data. Modula: Bandung.