Praktikum 3

Senarai Berantai Dua Arah (Double Linked List)

POKOK BAHASAN:

- ✓ Konsep double linked list
- ✓ Struktur double linked list
- ✓ Implementasi double linked list dalam Bahasa C

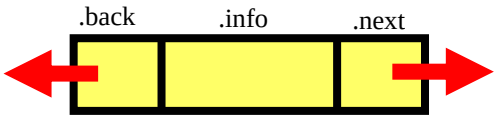
TUJUAN BELAJAR:

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami pengertian *double linked list*, gunanya dan dapat mengimplementasikan dalam pemrograman
- ✓ Memahami logika operasi-operasi yang ada dalam *double linked list*, dan dapat menerapkan dalam bentuk program
- ✓ Mengidentifikasi permasalahan-permasalahan pemrograman yang harus diselesaikan dengan menggunakan *double linked list*, sekaligus menyelesaikannya

DASAR TEORI:

Double linked list dibentuk dengan menyusun sejumlah elemen sehingga pointer *next* menunjuk ke elemen yang mengikutinya dan pointer *back* menunjuk ke elemen yang mendahuluinya. Dalam gambar 3.1 ini diilustrasikan sebuah simpul dalam *double linked list.* Info adalah data yang digunakan dalam simpul, back adalah pointer yang menunjuk pada simpul sebelumnya, dan next adalah pointer yang menunjuk pada simpul sesudahnya



Gambar 3.1 Ilustrasi sebuah simpul dalam Double Linked List

1. DEKLARASI SIMPUL

Cara mendeklarasikan sebuah simpul dalam double linked list adalah sebagai berikut:

```
struct DoubleLinkNode
{
   int                        bil;
   struct DoubleLinkNode   *back;
   struct DoubleLinkNode   *next;
};
struct DoubleLinkNodeRec *Node;
```

2. OPERASI DALAM DOUBLE LINKED LIST

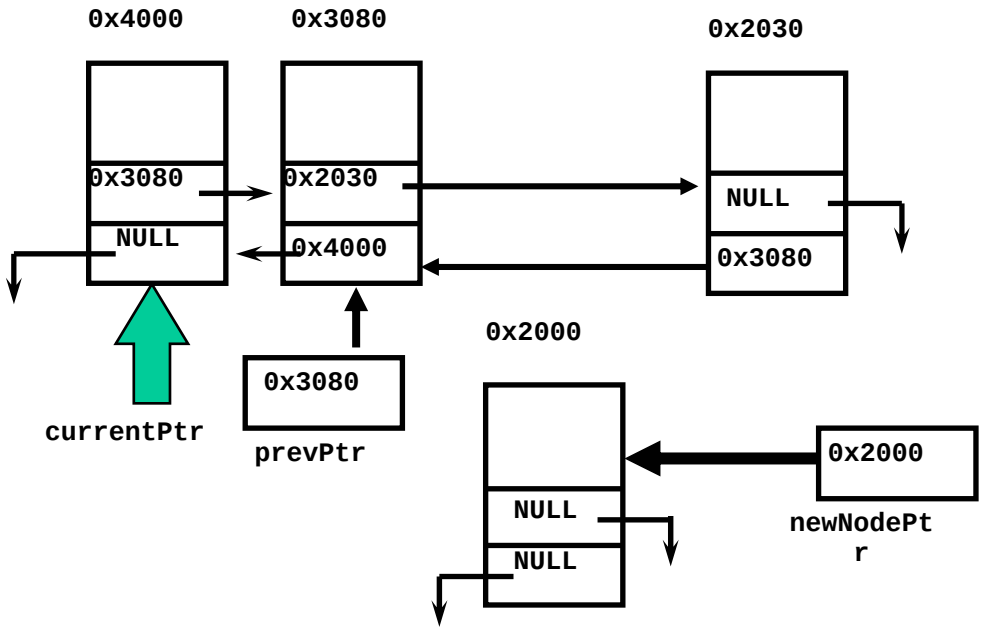
Sama seperti operasi yang ada dalam Single Linked List, operasi yang ada dalam Double Linked List dibagi menjadi 2, yaitu menyisipkan simpul dan menghapus simpul. Untuk menyisipkan simpul, secara garis besar juga sama seperti dalam Single Linked List. Namun untuk memahami konsepnya, kali ini kita membahas 2 macam operasi, yaitu menyisipkan sebagai simpul terakhir dan menyisipkan simpul di tengah.

2.1 OPERASI PENYISIPAN SIMPUL DI TENGAH

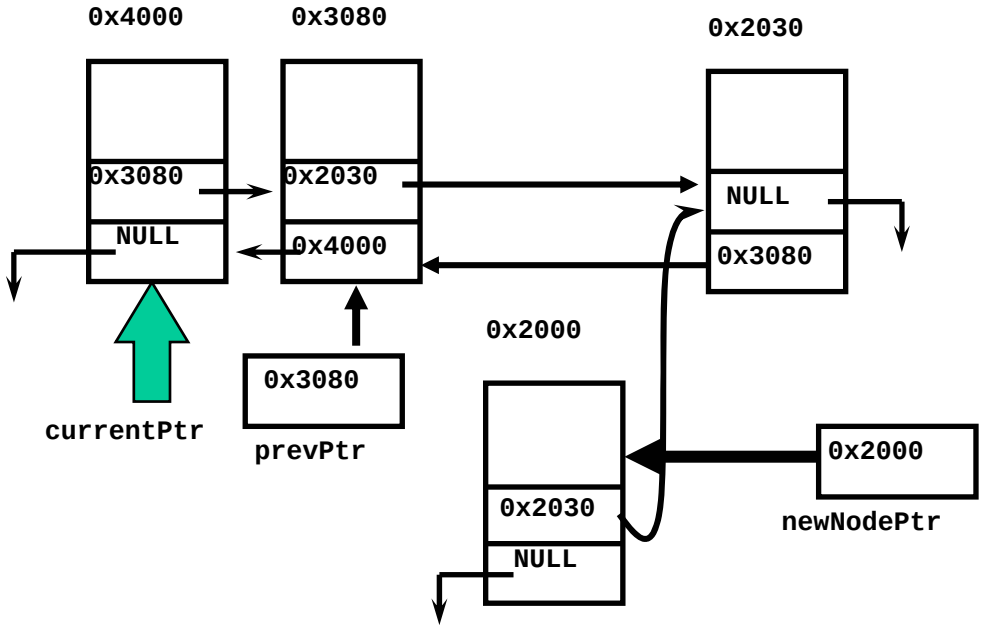
Dalam Gambar 3.2 diilustrasikan langkah-langkah untuk menyisipkan simpul di tengah.

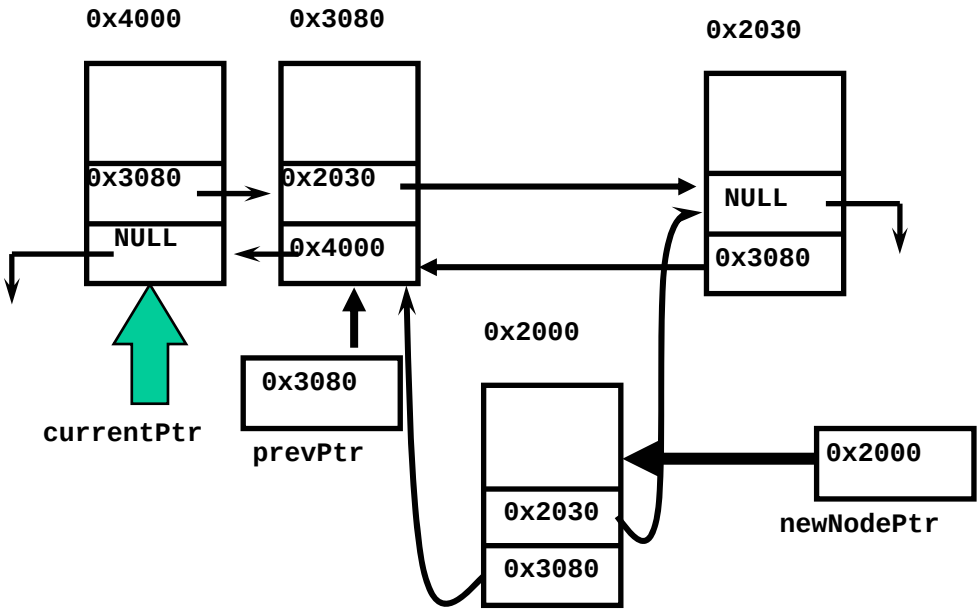
Langkah-langkah untuk Menyisipkan Simpul di Tengah adalah sebagai berikut:

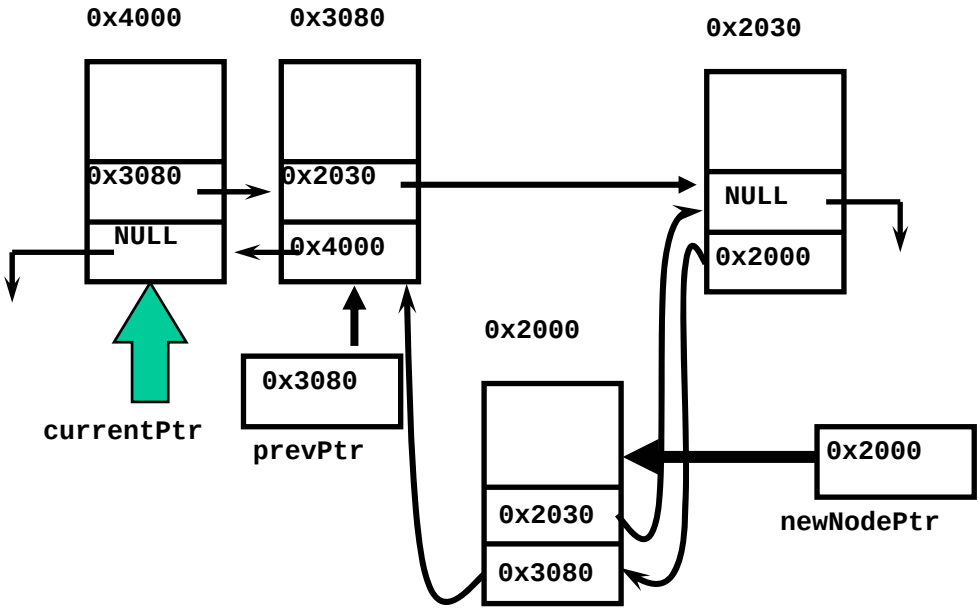
1. `newNodePtr` adalah simpul yang akan disisipkan, `prevPtr` adalah simpul yang akan disisipkan sesudahnya.
2. Gerakkan pointer `next` dari `newNodePtr` pada `prevPtr->next`.
3. Gerakkan pointer `back` dari `newNodePtr` pada `prevPtr`.
4. Arahkan `prevPtr->next->back` pada `newNodePtr`.
5. Arahkan pointer `next` pada `prevPtr` pada `newNodePtr`.

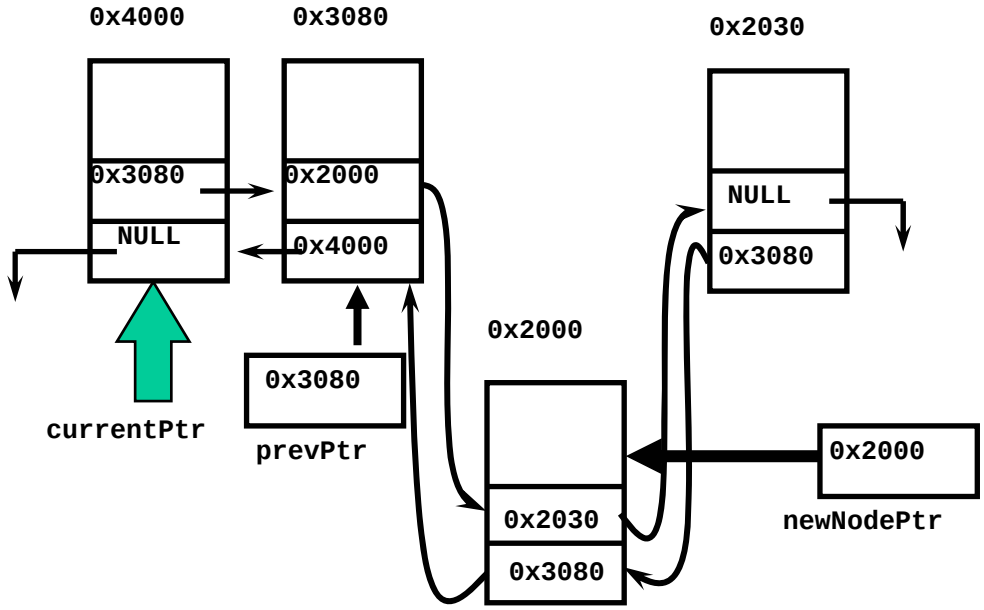


Gambar 3.2 Ilustrasi *Double Linked List* sebelum Disisipi pada Posisi Tengah




0x4000
0x3080
0x2030
0x3080
NULL
0x2030
0x4000
NULL
0x3080
0x2000
0x3080
currentPtr
prevPtr
0x2000
0x2030
0x3080
newNodePtr

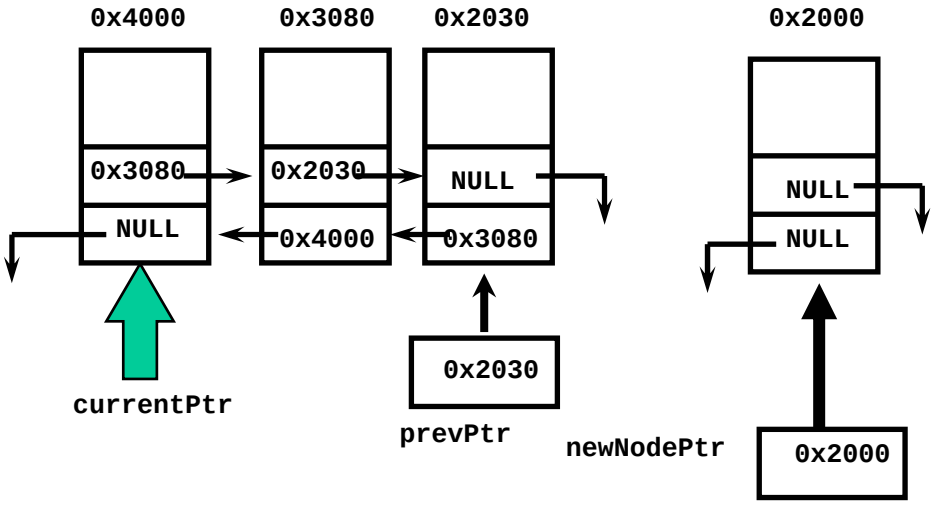

0x4000
0x3080
0x2030
0x3080
NULL
0x2030
0x4000
NULL
0x2000
0x2000
0x3080
currentPtr
prevPtr
0x2000
0x2030
0x3080
newNodePtr



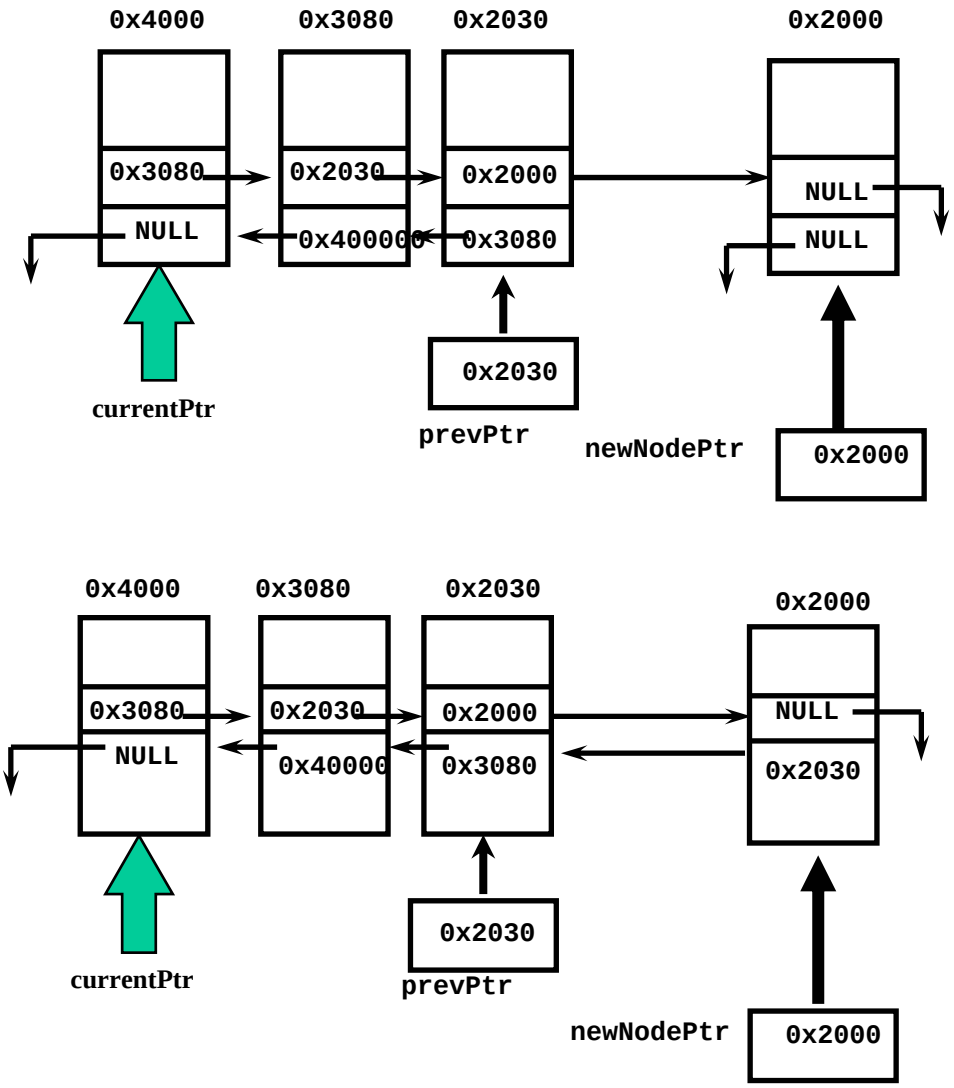
Gambar 3.3 Ilustrasi Proses Penyisipan sebagai Simpul di Tengah pada *Double Linked List*

2.2 OPERASI PENYISIPAN SIMPUL DI AKHIR

Dalam Gambar 3.2 diilustrasikan langkah-langkah untuk menyisipkan simpul di tengah.



Gambar 3.4 Ilustrasi *Double Linked List* sebelum Disisipi pada Simpul di Akhir



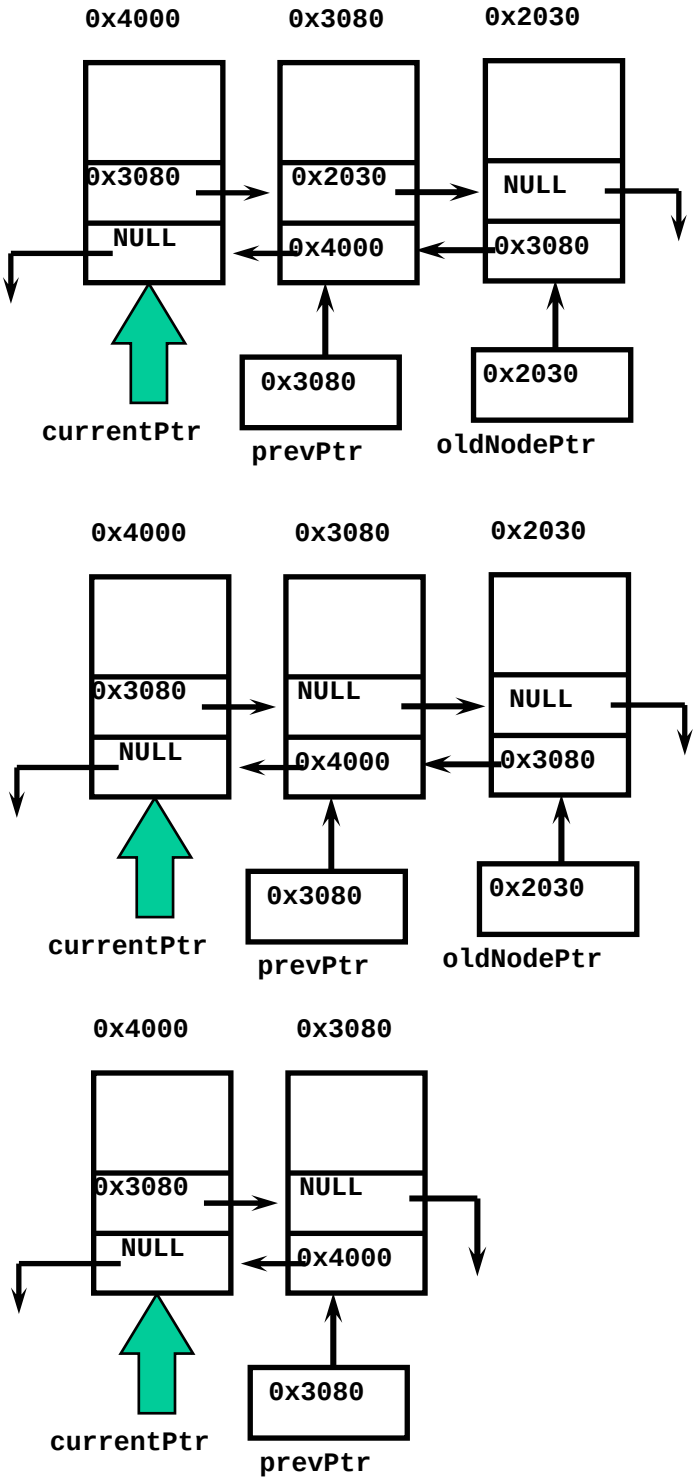
Gambar 3.5 Ilustrasi Proses Penyisipan sebagai Simpul Terakhir pada *Double Linked List*

Langkah-langkah untuk Menyisipkan Simpul sebagai Simpul Terakhir di atas adalah sebagai berikut:

1. `newNodePtr` adalah simpul yang akan disisipkan
2. Gerakkan pointer `prevPtr` yang semula menunjuk pada simpul `head` hingga ketemu dengan simpul yang pointer `next`-nya menunjuk pada `NULL`
3. Arahkan pointer `next` pada `prevPtr` pada `newNodePtr`
4. Arahkan pointer `back` pada `newNodePtr` ke arah `prevPtr`

2.3 OPERASI PENGHAPUSAN DI AKHIR

Pada gambar 3.6 dapat dilihat ilustrasi dari penghapusan simpul di akhir.



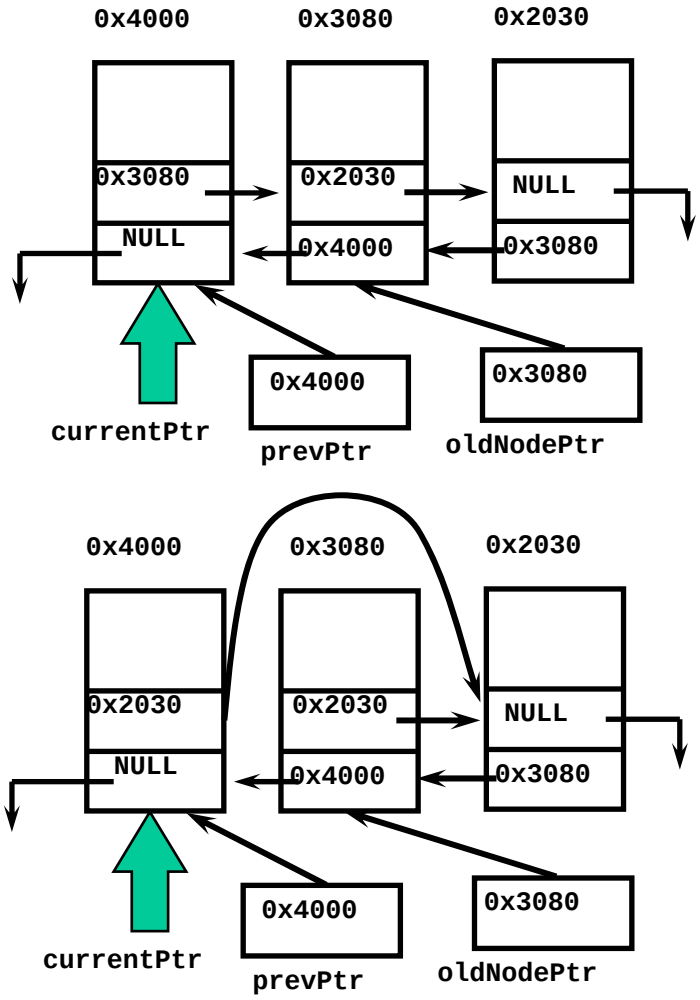
Gambar 3.6 Ilustrasi Operasi Penghapusan Simpul di Akhir

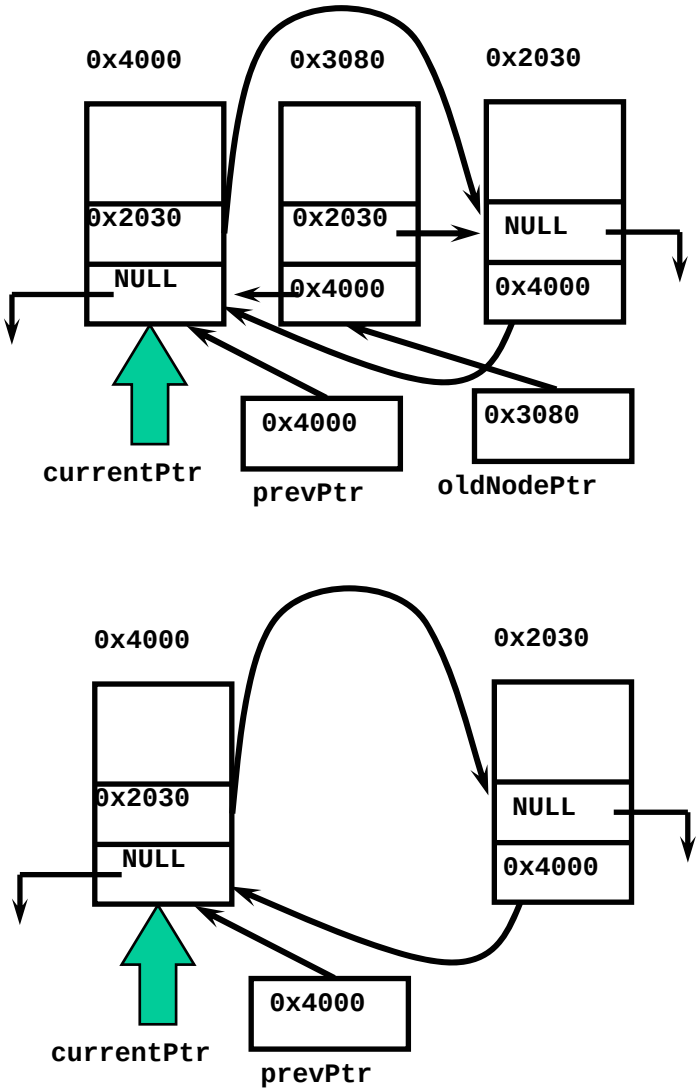
Langkah-langkah untuk Menghapus Simpul di Akhir adalah sebagai berikut:

1. `oldNodePtr` adalah simpul yang akan dihapus, `prevPtr` adalah simpul yang sebelumnya.
2. Arahkan `prevPtr->next` ke NULL.
3. Hapus `oldNodePtr`.

2.4 OPERASI PENGHAPUSAN DI TENGAH

Pada gambar 3.7 dapat dilihat ilustrasi dari penghapusan simpul di tengah.





Gambar 3.7 Ilustrasi Operasi Penghapusan Simpul di Tengah

Langkah-langkah untuk Menghapus Simpul di Tengah adalah sebagai berikut:

1. `oldNodePtr` adalah simpul yang akan dihapus, `prevPtr` adalah simpul yang sebelumnya.
2. Arahkan `prevPtr->next` ke `oldNodePtr->next`.
3. Arahkan `oldNodePtr->next->back` ke `oldNpdePtr->back`.
4. Hapus `oldNodePtr`.

TUGAS PENDAHULUAN:

1. Buatlah flowchart untuk operasi Membuat Double Linked List yang baru
2. Buatlah flowchart untuk menyisipkan simpul, baik simpul di awal, di tengah maupun di akhir dari linked list.
3. Buatlah flowchart untuk menghapus simpul, baik simpul di awal, di tengah maupun di akhir dari linked list.

PERCOBAAN:

1. Buatlah workspace untuk praktikum Struktur Data dengan menggunakan Visual C++.
2. Buatlah project untuk praktikum KETIGA
3. Cobalah untuk masing-masing percobaan di bawah ini dengan menambahkan program utamanya.
4. Selesaikan soal-soal yang ada dengan mengimplementasikan flowchart yang anda buat pada Tugas Pendahuluan.

Percobaan 1 : Fungsi Membentuk Double Linked List

```
//deklarasi dari simpul
struct DoubleLinkedNode
{
      int bil;
      struct DoubleLinkedNode *back;
      struct DoubleLinkedNode *next;
};
struct DoubleLinkedNode *head, *tail;

void bentuk_awal()
{
      struct DoubleLinkedNode *awal;
      int j=0;char jawab[2];
      while(1)
      {
            awal=(struct  DubleLinkedNode*)  malloc(sizeof  (struct
            DoubleLinkedNode));
            printf("Masukkan bilangan :");
            scanf("%d",&awal->bil);
```

```
            if(j==0)
            {
                  awal->next=NULL;
                  awal->back=NULL;
                  head = awal;
                  tail = awal;
            }
            else
            {
                  tail->next=awal;
                  awal->next=NULL;
                  awal->back=tail;
                  tail = awal;
            }

            printf("Ada data lagi(y/t):"); scanf("%s", &jawab);

      if((strcmp(jawab,"Y")==0)||(strcmp(jawab,"y")==0))
            {
                  j++;continue;
            }
      else if ((strcmp(jawab,"T")==0) || (strcmp(jawab,"t") == 0))
                  break;
      }
}
```

Percobaan 2 : Fungsi untuk Menampilkan Double Linked List dengan metode LIFO

```
void tampil_list_lifo()
{
      struct DoubleLinkedNode *lifo;
      printf("Data Bilangan yang Telah Diinputkan secara LIFO :\n");
      lifo = tail;
      while(lifo!=NULL)
      {
            printf("%d\t",lifo->bil);
            lifo=lifo->back;
      }
      printf("\n");
}
```

Percobaan 3 : Fungsi untuk Menampilkan Double Linked List dengan Metode FIFO

```
void tampil_list_fifo()
{

      struct DoubleLinkedNode *fifo;
      printf("Data Bilangan yang Telah Diinputkan secara FIFO :\n");
      fifo = head;
```

```
      while(fifo!=NULL)
      {
            printf("%d\t",fifo->bil);
            fifo=fifo->next;
      }
      printf("\n");
}
```

Percobaan 4 : Fungsi untuk Menyisipkan Simpul di Tengah

```
Percobaan 4 : Fungsi untuk Menyisipkan Simpul di Tengah

void sisip_simpul_tengah()
{
      int num;
      struct DoubleLinkedNode *sisip, *stl, *sbl;
      sisip=(struct           DoubleLinkedNode*)malloc(sizeof(struct
      DoubleLinkedNode));
      printf("Tuliskan bilangan yang akan disisipkan : ");
      scanf("%d",&sisip->bil);
      sisip->next=NULL;
      sisip->back=NULL;
      printf("Bilangan disisipkan sebelum data : ");
      scanf("%d",&num);
      stl = head;
      do {
            sbl = stl;
            stl = stl->next;
      }
      while (stl->bil!=num);
      sisip->next= stl;
      sisip->back= sbl;
      sbl->next= sisip;
      stl->back= sisip;
}
```

Percobaan 5 : Fungsi untuk Menghapus Simpul Tertentu

```
void hapus_simpul()
{
      int num;
      struct DoubleLinkedNode *sbl, *hapus, *stl;

      printf("Masukkan data yang akan dihapus : ");
      scanf("%d",&num);
      hapus = head;

      //Menghapus Simpul Awal
      if (hapus->bil == num)
      {
            head = head->next;
            head->back = NULL;
            free(hapus);
      }
```

```
      else
      {
            do{
                  sbl = hapus;
                  hapus = hapus->next;}
            while (hapus->bil!=num);

            //Menghapus Simpul Terakhir
            if (hapus->next == NULL)
            {
                  sbl->next=NULL;
                  tail = sbl;
                  free(hapus);
            }

            //Menghapus Simpul di Tengah
            else
            {
                  stl = hapus->next;
                  sbl->next = stl;
                  stl->back = sbl;
                  free(hapus);
            }
      }
}
```

LATIHAN:

1. Buatlah sebuah program yang mengimplementasikan Double Linked List, dimana data yang dipakai adalah data buku yang ada dalam sebuah perpustakaan (judul, nama pengarang, penerbit). Program juga mengimplementasikan penambahan dan pengurangan simpul pada Linked List berdasarkan judul buku.
2. Implementasikan Double Linked List ini juga untuk memvisualisasikan antrian mobil (menampilkan dengan metode FIFO seperti percobaan di atas) yang ada pada sebuah perparkiran. Data yang digunakan adalah : no plat nomor, merk mobil, nama pemilik. Data ditambahkan dan dikurangi berdasarkan no plat nomor.