# DOUBLE LINKED LIST

**DOUBLE LINKED LIST**

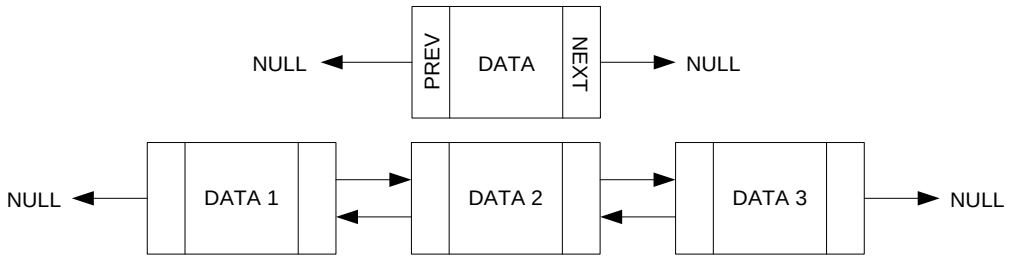
- Pada dasarnya, penggunaan Double Linked List hampir sama dengan penggunaan Single Linked List yang telah kita pelajari pada materi sebelumnya. Hanya saja Double Linked List menerapkan sebuah pointer baru, yaitu **prev**, yang digunakan untuk menggeser mundur selain tetap mempertahankan pointer **next**.
- Keberadaan 2 pointer penunjuk (**next** dan **prev**) menjadikan Double Linked List menjadi lebih fleksibel dibandingkan Single Linked List, namun dengan mengorbankan adanya memori tambahan dengan adanya pointer tambahan tersebut.
- Ada 2 jenis Double Linked List, yaitu: Double Linked List Non Circular dan Double Linked List Circular.

## I. DOUBLE LINKED LIST NON CIRCULAR (DLLNC)

### a. DLLNC

- DLLNC adalah sebuah Linked List yang terdiri dari dua arah pointer, dengan node yang saling terhubung, namun kedua pointernya menunjuk ke NULL.
- Setiap node pada linked list mempunyai field yang berisi data dan pointer yang saling berhubungan dengan node yang lainnya.

### b. GAMBARAN NODE DLLNC

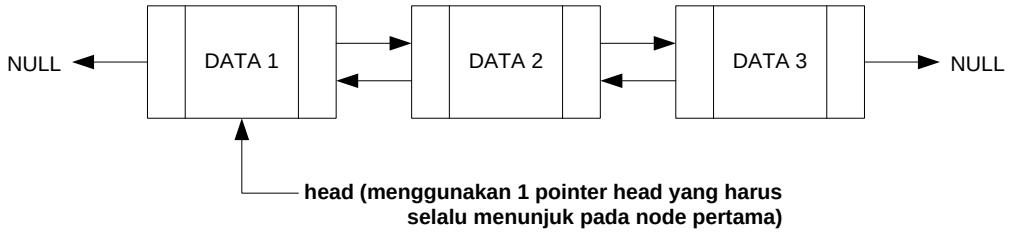



### c. PEMBUATAN DLLNC

- **Deklarasi Node**

```
typedef struct TNode
{
    int data;
    TNode *next;
    TNode *prev;
}
```

- **Pembuatan DLLNC dengan Head**
  - Ilustrasi :




- Fungsi-fungsi yang biasa digunakan :
  - ✓ Fungsi untuk inisialisasi awal

```
void init()                                          // inisialisasi awal
{
      TNode *head;
      head = NULL;
}
```

  - ✓ Perlu diperhatikan :
    - Fungsi ini harus ada, untuk memunculkan node awal.
    - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini. Jangan lupa untuk mendeklarasikan node-nya terlebih dahulu.

  - ✓ Fungsi untuk mengecek kosong tidaknya Linked List

```
int isEmpty()                        // mengecek kosong tidaknya Linked List
{
      if(head == NULL)
            return 1;
      else
            return 0;
}
```

  - ✓ Perlu diperhatikan :
    - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

  - ✓ Fungsi untuk menambahkan data di depan

```
void insertDepan(int value)                    // penambahan data di depan
{
      TNode *baru;
      baru = new TNode;                              // pembentukan node baru

      baru->data = value;             // pemberian nilai terhadap data baru
      baru->next = NULL;             // data pertama harus menunjuk ke NULL
      baru->prev = NULL;             // data pertama harus menunjuk ke NULL

      if(isEmpty() == 1)                           // jika Linked List kosong
      {
          head = baru;                 // head harus selalu berada di depan
          head->next = NULL;
          head->prev = NULL;
      }
      else                            // jika Linked List sudah ada datanya
      {
          baru->next = head;                // node baru dihubungkan ke head
          head->prev = baru;           // node head dihubungkan ke node baru
```

```
            head = baru;                    // head harus selalu berada di depan
        }

        printf("data masuk\n");
}
```

- ✓ Perlu diperhatikan :
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di depan. Misalkan saja data pada Linked List ada 4.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menambahkan data di belakang

```
void insertBelakang(int value)                  // penambahan data di belakang
{
        TNode *baru, *bantu;
        baru = new TNode;                               // pembentukan node baru

        baru->data = value;             // pemberian nilai terhadap data baru
        baru->next = NULL;             // data pertama harus menunjuk ke NULL
        baru->prev = NULL;             // data pertama harus menunjuk ke NULL

        if(isEmpty() == 1)                          // jika Linked List kosong
        {
            head = baru;                    //head harus selalu berada di depan
            head->next = NULL;
            head->prev = NULL;
        }
        else
        {
            bantu = head;                       // bantu diletakan di head dulu
            while(bantu->next != NULL)
            {
                 bantu = bantu->next         // menggeser hingga node terakhir
            }

            baru->next = baru;                // node baru dihubungkan ke head
            head->prev = bantu;          // node head dihubungkan ke node baru
        }
        printf("data masuk\n");
}
```

- ✓ Perlu diperhatikan :
  - Jika Linked List hanya menggunakan head, maka dibutuhkan satu pointer untuk membantu mengetahui node terakhir dari Linked List. Dalam code di atas digunakan pointer bantu.
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di belakang. Misalkan saja data pada Linked List ada 4.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

✓ Fungsi untuk menambahkan data di tengah (menyisipkan data)

```
void insertTengah(int value, int cari)          //penambahan data di tengah
{
      TNode *baru, *bantu, *bantu2;
      baru = new TNode;                              // pembentukan node baru

      baru->data = value;             // pemberian nilai terhadap data baru
      baru->next = NULL;             // data pertama harus menunjuk ke NULL
      baru->prev = NULL;             // data pertama harus menunjuk ke NULL
      bantu = head;                          // bantu diletakan di head dulu

      while(bantu->data != cari)
      {
          bantu = bantu->next;          //menggeser hingga didapat data cari
      }

      bantu2 = bantu->next;    // menghubungkan ke node setelah yang dicari
      baru->next = bantu2;                         // menghubungkan node baru
      bantu2->prev = baru;
      bantu->next = baru;      // menghubungkan ke node sebelum yang dicari
      baru->prev = bantu;
}
```

✓ Perlu diperhatikan :

- Dibutuhkan satu pointer untuk membantu mencari node di mana data yang ingin disisipkan ditempatkan. Dalam code di atas digunakan pointer bantu.
- Penggunaan pointer bantu2 pada code di atas sebenarnya bisa digantikan dengan pemanfaatan pointer bantu. Bagaimana caranya?
- Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di tengah. Misalkan saja data pada Linked List ada 4, lalu sisipkan data baru setelah node kedua.
- Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

✓ Fungsi untuk menghapus data di depan

```
void deleteDepan()                                  // penghapusan data di depan
{
      TNode *hapus;

      if(isEmpty() == 0)                              // jika data belum kosong
      {
          if(head->next != NULL)                // jika data masih lebih dari 1
          {
               hapus = head;                         // letakan hapus pada head
               head = head->next;  // menggeser head (karena head harus ada)
               head->prev = NULL;                 // head harus menuju ke NULL
               delete hapus; //proses delete tidak boleh dilakukan jika node
masih ditunjuk oleh pointer
          }
          else                                        // jika data tinggal head
          {
               head = NULL;                 // langsung diberi nilai NULL saja
```

```
          }
          printf("data terhapus\n");
      }
      else                                          // jika data sudah kosong
          printf("data kosong\n");
}
```

- ✓ Perlu diperhatikan :
  - Dibutuhkan satu pointer untuk membantu memindahkan head ke node berikutnya. Dalam code di atas digunakan pointer hapus. Mengapa head harus dipindahkan?
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di depan. Misalkan saja data pada Linked List ada 4.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menghapus data di belakang

```
void deleteBelakang()                          // penghapusan data di belakang
{
      TNode *hapus;

      if(isEmpty() == 0)                              // jika data belum kosong
      {
          if(head->next != NULL)              // jika data masih lebih dari 1
          {
               hapus = head;                         // letakan hapus pada head
               while(hapus->next != NULL)
               {
                    hapus = hapus->next;       // menggeser hingga node akhir
               }

               hapus->prev->next = NULL;    // menghubungkan node sebelumnya
dengan NULL
               delete hapus; //proses delete tidak boleh dilakukan jika node
sedang ditunjuk oleh pointer
          }
          else                                          // jika data tinggal head
          {
              head = NULL;                // langsung diberi nilai NULL saja
          }
          printf("data terhapus\n");
      }
      else                                          // jika data sudah kosong
          printf("data kosong\n");

}
```

- ✓ Perlu diperhatikan :
  - Jika Linked List hanya menggunakan head, maka dibutuhkan satu pointer untuk membantu mengetahui node terakhir dari Linked List. Dalam code di atas digunakan pointer hapus.
  - Jangan lupa untuk tetap mengaitkan node terakhir ke NULL.

- Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di belakang. Misalkan saja data pada Linked List ada 4.
- Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

✓ Fungsi untuk menghapus data di tengah

```
void deleteTengah(int cari)                    // penghapusan data di tengah
{
      TNode *hapus, *bantu, *bantu2;

      hapus = head;                                  // letakan hapus pada head
      while(hapus->data != cari)
      {
          hapus = hapus->next;                 // menggeser hingga data cari
      }
      bantu2 = hapus->next;      // mengkaitkan node sebelum dan sesudahnya
      bantu = hapus->prev;
      bantu->next = bantu2;
      bantu2->prev = bantu;
      printf("data terhapus\n");
      delete hapus; //proses delete tidak boleh dilakukan jika node sedang
ditunjuk oleh pointer
}
```

✓ Perlu diperhatikan :

- Dibutuhkan satu pointer untuk membantu mencari node di mana data yang ingin dihapus ditempatkan. Dalam code di atas digunakan pointer hapus.
- Penggunaan pointer bantu dan bantu2 pada code di atas sebenarnya bisa digantikan dengan pemanfaatan pointer hapus. Bagaimana caranya?
- Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di tengah. Misalkan saja data pada Linked List ada 4, lalu hapus data pada node ketiga.
- Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

✓ Fungsi untuk menghapus semua data

```
void clear()                                        // penghapusan semua data
{
      TNode *bantu, *hapus;
      bantu = head;                                 // letakan bantu pada head
      while (bantu != NULL)                        // geser bantu hingga akhir
      {
          hapus = bantu;
          bantu = bantu->next;
          delete hapus;                            // delete satu persatu node
      }

      head = NULL;         // jika sudah habis berikan nilai NULL pada head
```

```
}
```

- ✓ Perlu diperhatikan :
  - Dibutuhkan dua pointer untuk membantu menggeser dan menghapus, di mana dalam code di atas digunakan pointer bantu dan hapus.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

- ✓ Fungsi untuk menampilkan semua data

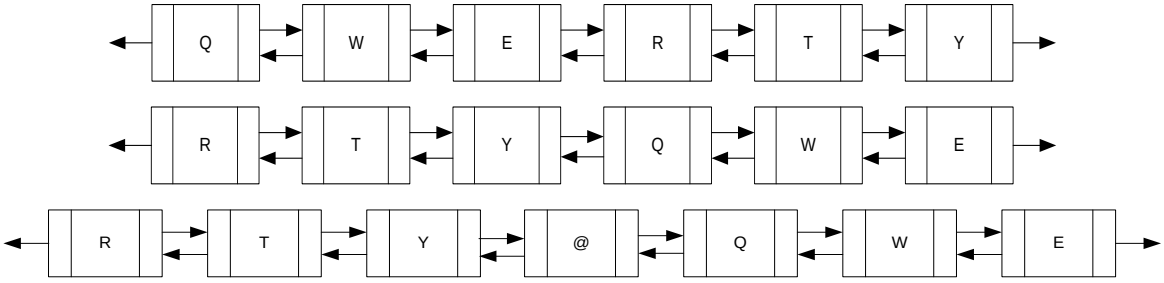
```
void cetak()                                        // menampilkan semua data
{
      TNode *bantu;
      bantu = head;                                 // letakan bantu pada head

      if(isEmpty() == 0)
      {
          while (bantu != NULL)
          {
              printf("%d ", bantu->data);     // cetak data pada setiap node
              bantu = bantu->next;               // geser bantu hingga akhir
          }
          printf("\n");
      }
      else                                           // jika data sudah kosong
          printf("data kosong");
}
```

- ✓ Perlu diperhatikan :
  - Dibutuhkan satu pointer untuk membantu menggeser, di mana dalam code di atas digunakan pointer bantu.
  - Setelah memahami penggunaan fungsi ini, bukalah Turbo C++ Anda, dan copy-kan fungsi ini.

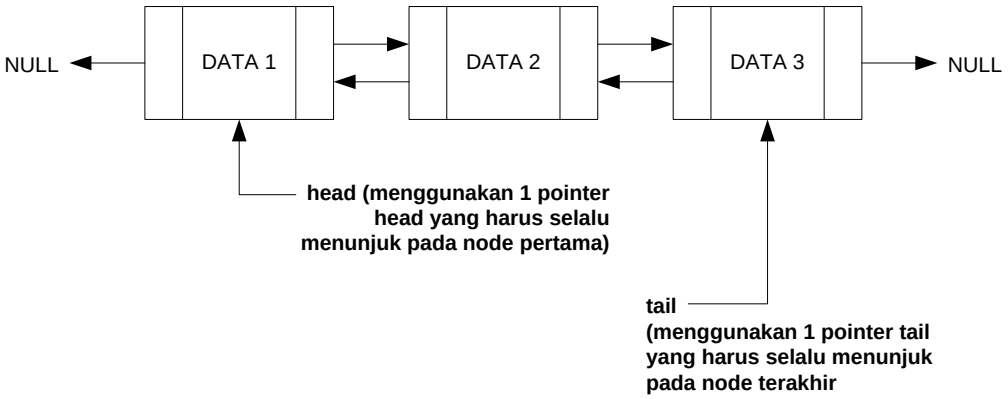
- **Latihan 1 :**
  1. Setelah deklarasi node dilakukan, dan semua fungsi sudah tersedia. Sekarang gabungkan setiap fungsi yang ada pada sebuah program penuh dengan spesifikasi :
     - Pada program utama (main) berisi sebuah menu yang berisi fitur-fitur yang terdapat dari setiap fungsi yang sudah ada ada sebelumnya, yaitu : tambah data, hapus data, cek data kosong, dan cetak semua data.
     - Pada struct hanya terdapat 1 tipe data saja yaitu integer.
     - Sesuaikan fungsi-fungsi yang ada dengan program yang Anda buat (jangan langsung copy-paste dan digunakan).
  2. Buat program untuk enkripsi dan dekripsi password yang memanfaatkan Linked List, dengan spesifikasi :
     - Panjang password minimal 6 digit.
     - Isi password terserah dari user dan password diinputkan terlebih dahulu sebelumnya (penambahan data di belakang).

- Enkripsi dilakukan dengan memindahkan 3 node terakhir, menjadi node terdepan. Kemudian sisipkan 1 karakter baru (kunci) setelah node ketiga dari yang dipindahkan tersebut.
- Ilustrasi :




- Lakukan juga proses dekripsi-nya.
- Berikan juga fitur untuk menampilkan password.

- **Pembuatan DLLNC dengan Head dan Tail**
  - Ilustrasi :




  - Sebenarnya sama saja dengan DLLNC dengan menggunakan head saja, hanya saja perbedaannya terletak pada kemudahan pengaksesan di bagian data terakhirnya saja (tail).
  - Fungsi-fungsi yang biasa digunakan :
    - ✓ Fungsi untuk inisialisasi awal

```
void init()                                          //inisialisasi awal
{
      TNode *head, *tail;
      head = NULL;
      tail = NULL;
}
```

    - ✓ Fungsi untuk mengecek kosong tidaknya Linked List

```
int isEmpty()                          //mengecek kosong tidaknya Linked List
{
      if(tail == NULL)
           return 1;
      else
           return 0;
}
```

- ✓ Fungsi untuk menambahkan data di depan

```
void insertDepan(int value)                          //penambahan data di depan
{
      TNode *baru;
      baru = new TNode;                                 // pembentukan node baru

      baru->data = value;              // pemberian nilai terhadap data baru
      baru->next = NULL;              // data pertama harus menunjuk ke NULL
      baru->prev = NULL;              // data pertama harus menunjuk ke NULL

      if(isEmpty() == 1)                              // jika Linked List kosong
      {
          head = baru;                                 // letakan baru pada head
          tail = head;                                                // head = tail
          head->next = NULL;
          head->prev = NULL;
          tail->next = NULL;
          tail->prev = NULL;
      }
      else                               // jika Linked List sudah ada datanya
      {
          baru->next = head;     // menyambungkan data baru dengan head lama
          head->prev = baru;
          head = baru;                    //head harus selalu berada di depan
      }
      printf("data masuk\n");
}
```

- ✓ Perlu diperhatikan :
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di depan.

- ✓ Fungsi untuk menambahkan data di belakang

```
void insertBelakang(int value)                  //penambahan data di belakang
{
      TNode *baru;
      baru = new TNode;                                 // pembentukan node baru

      baru->data = value;              // pemberian nilai terhadap data baru
      baru->next = NULL;              // data pertama harus menunjuk ke NULL
      baru->prev = NULL;              // data pertama harus menunjuk ke NULL

      if(isEmpty() == 1)                              // jika Linked List kosong
      {
          head = baru;                                 // letakan baru pada head
          tail = head;                                                // head = tail
          head->next = NULL;
          head->prev = NULL;
          tail->next = NULL;
          tail->prev = NULL;
      }
      else                               // jika Linked List sudah ada datanya
      {
          tail->next = baru;     // menghubungkan data baru dengan tail lama
          baru->prev = tail;
          tail = baru;                // tail harus selalu berada di belakang
          tail->next = NULL;
```

```
      }
      printf("data masuk\n");
}
```

- ✓ Perlu diperhatikan :
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di belakang.
  - Perhatikan kemudahan pengaksesan node terakhirnya.

- ✓ Fungsi untuk menambahkan data di tengah (menyisipkan data)

```
void insertTengah(int value, int cari)          //penambahan data di tengah
{
      TNode *baru, *bantu, *bantu2;
      baru = new TNode;                                 // pembentukan node baru
      baru->data = value;             // pemberian nilai terhadap data baru
      baru->next = NULL;             // data pertama harus menunjuk ke NULL
      baru->prev = NULL;             // data pertama harus menunjuk ke NULL

      bantu = head;                                     // letakan bantu di awal
      while(bantu->data != cari)
      {
          bantu = bantu->next;                // geser bantu sampai data cari
      }

      bantu2 = bantu->next;    // menghubungkan ke node setelah yang dicari
      baru->next = bantu2;                          // menghubungkan node baru
      bantu2->prev = baru;
      bantu->next = baru;      // menghubungkan ke node sebelum yang dicari
      baru->prev = bantu;
}
```

- ✓ Perlu diperhatikan :
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penambahan di tengah. Misalkan saja data pada Linked List ada 4, lalu sisipkan data sebelum node ketiga.

- ✓ Fungsi untuk menghapus data di depan

```
void deleteDepan()                                  // penghapusan data di depan
{
      TNode *hapus;

      if(isEmpty() == 0)                              // jika data belum kosong
      {
          if(head->next != NULL)                  // jika data tidak tinggal 1
          {
            hapus = head;                           // letakan hapus pada head
            head = head->next;    // menggeser head (karena head harus ada)
            head->prev = NULL;                      // head harus menuju ke NULL
            delete hapus;   //proses delete tidak boleh dilakukan jika node
sedang ditunjuk oleh pointer
          }
          else                                      // jika data tinggal head
          {
              head = NULL;                          // langsung diberi NULL saja
              tail = NULL;
```

```
          }
          printf("data terhapus\n");
      }
      else                                                  // jika data kosong
          printf("data kosong\n");
}
```

- ✓ Perlu diperhatikan :
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di depan.

- ✓ Fungsi untuk menghapus data di belakang

```
void deleteBelakang()                          // penghapusan data di belakang
{
      TNode *hapus;

      if(isEmpty() == 0)                             // jika data belum kosong
      {
          if(head ->next != NULL)                // jika data tidak tinggal 1
          {
            hapus = tail;                          // letakan hapus pada tail
            tail = tail->prev;                                // menggeser tail
            tail->next = NULL;                   // tail harus menuju ke NULL
            delete hapus;   //proses delete tidak boleh dilakukan jika node
sedang ditunjuk oleh pointer
          }
          else // jika data tinggal head
          {
              head = NULL;                       // langsung diberi NULL saja
              tail = NULL;
          }
          printf("data terhapus\n");
      }
      else                                                  // jika data kosong
          printf("data kosong\n");
}
```

- ✓ Perlu diperhatikan :
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di belakang.
  - Perhatikan kemudahan pengaksesan node terakhirnya.

- ✓ Fungsi untuk menghapus data di tengah

```
void deleteTengah(int cari)                    // penghapusan data di tengah
{
      TNode *hapus, *bantu, *bantu2;

      hapus = head;                                  // letakan hapus pada head
      while(hapus->data != cari)
      {
          hapus = hapus->next;                 // menggeser hingga data cari
      }
      bantu2 = hapus->next;      // mengkaitkan node sebelum dan sesudahnya
      bantu = hapus->prev;
      bantu->next = bantu2;
      bantu2->prev = bantu;
      printf("data terhapus\n");
```

```
      delete hapus; //proses delete tidak boleh dilakukan jika node sedang
ditunjuk oleh pointer
}
```

- ✓ Perlu diperhatikan :
  - Baca code beserta panduan proses yang terjadi, pahami, lalu gambarkan ilustrasi proses terjadinya penghapusan di tengah. Misalkan saja data pada Linked List ada 4, lalu hapus data pada node ketiga.

- ✓ Fungsi untuk menghapus semua data

```
void clear()                                        // penghapusan semua data
{
      TNode *bantu, *hapus;
      bantu = head;                                  // letakan bantu pada head
      while (bantu != NULL)                         // geser bantu hingga akhir
      {
          hapus = bantu;
          bantu = bantu->next;
          delete hapus;                             // delete satu persatu node
      }

      head = NULL;         // jika sudah habis berikan nilai NULL pada head
      tail = NULL;         // jika sudah habis berikan nilai NULL pada tail
}
```

- ✓ Fungsi untuk menampilkan semua data

```
void cetak()                                        // menampilkan semua data
{
      TNode *bantu;
      bantu = head;                                  // letakan bantu pada head

      if(isEmpty() == 0)
      {
          while (bantu != tail->next)
          {
              printf("%d ", bantu->data);     // cetak data pada setiap node
              bantu = bantu->next;               // geser bantu hingga akhir
          }
          printf("\n");
      }
      else                                           // jika data sudah kosong
          printf("data kosong");
}
```

- **Latihan 2 :**
  1. Lakukan latihan 1 no 1 dengan DLLNC dengan head dan tail, bandingkan kemudahan penggunaannya.
  2. Lakukan latihan 1 no 2 dengan DLLNC dengan head dan tail, lalu berikan spesifikasi tambahan :
     - Tambahkan 2 macam user : anonymous dan admin.
     - Pada bagian fitur menampilkan password, jika admin user yang ingin melihat password tampilkan password secara benar (dari depan ke belakang),

namun jika anonymous user yang ingin melihat password tampilkan password secara terbalik (dari belakang ke depan – manfaatkan tail dalam pengaksesan semacam ini).

**d. TUGAS (DLLNC)**

1. Dari latihan 2 no 1 yang sudah Anda buat, buat juga fitur tambahan untuk melakukan sorting (ascending dan descending).
2. Dengan memanfaatkan semua fitu yang sudah ada sebelumnya. Buat sebuah aplikasi **SIMULASI FRAGMENTASI HARDDISK**. Dengan spesifikasi sebagai berikut :
   - Setiap node dilambangkan sebagai 1 fragmen harddisk dalam komputer Anda.
   - Dalam setiap fragmen berisi :
     - Nama data (unik).
     - Jenis data (aplikasi, OS, dokumen)
     - Ukuran data.
   - Terdapat menu untuk melakukan :
     - Penambahan fragmen / penambahan data.
     - Penghapusan fragmen / penghapusan data.
     - Pencarian fragmen (bisa ditambahkan fitur untuk mencari ukuran terbesar/terkecil, pencarian berdasarkan nama ataupun jenis data).
     - Edit isi fragmen.
     - Menampilkan isi fragmen (bisa dikombinasikan dengan fitur pencarian).
     - Melakukan defragmentasi (sorting), di mana bisa dilakukan terhadap : jenis data dan ukuran data.

## II. Double Linked List Cicular

1. Dengan Head
   - Menggunakan 1 pointer head
   - Head selalu menunjuk node pertama

Sebelumnya kita harus mendeklarasikan dulu pointer head :

```
TNode *head;
```

Setelah kita mendeklarasikan pointer head, kita belum bisa secara langsung mendeklarasikan node yang dituju. Sehingga pointer head harus dibuat bernilai *null* terlebih dahulu :

```
head = NULL;
```

untuk mengetahui apakah suatu Linked List kosong atau tidak, kita dapat mengetahuinya dengan mengecek nilai dari pointer Head-nya.

```
int isEmpty() {
      if(head==NULL) return 1;
      else return 0;
}
```

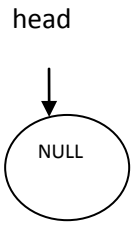
Contoh program :

- Penambahan di depan

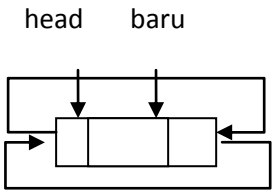
```
void tambahdata (int databaru){
   TNode *baru,*bantu;
   //pointer bantu digunakan untuk menunjuk node terakhir (head->prev)
   baru = new TNode;
   baru -> data = databaru;
   baru -> next = baru;
   baru -> prev = baru;

   if (isEmpty()==1) {
         head=baru;
         head->next=head;
         head->prev=head;
   }
   else {
         bantu=head->prev;
         baru->next=head;
         head->prev=baru;
         head=baru;
         head->prev=bantu;
         bantu->next=head;
   }
   printf("data masuk");
}
```

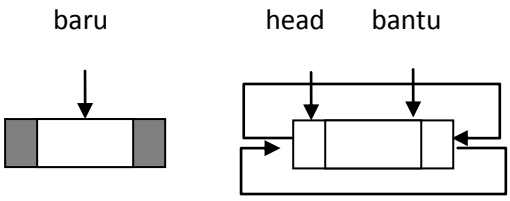
Penggambaran :

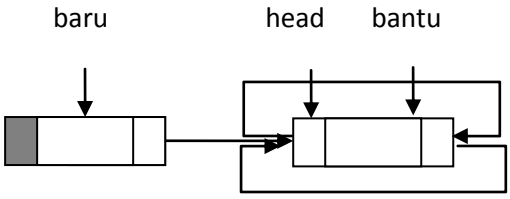


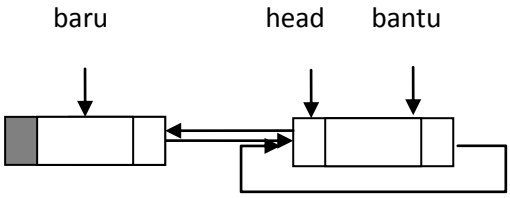

Setelah dibuat node baru dan jika diketahui head==NULL :

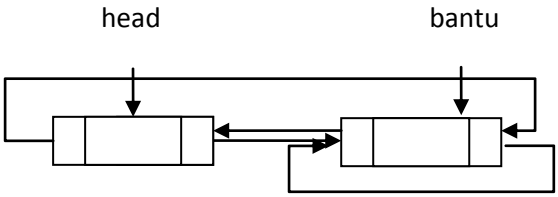


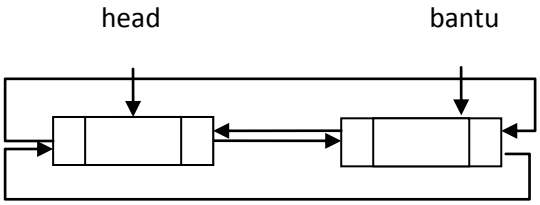

Bila kita membuat node baru lagi maka :






- Penambahan di belakang

```
void insertBelakang (int databaru){
   TNode *baru,*bantu;
   baru = new TNode;
   baru->data = databaru;
   baru->next = baru;
   baru->prev = baru;
   if(isEmpty()==1){
         head=baru;
         head->next = head;
         head->prev = head;
   }
   else {
         bantu=head->prev;
         bantu->next = baru;
         baru->prev = bantu;
         baru->next = head;
         head->prev = baru;
   }
   printf("data masuk");
}
```

- Tampil

```
void tampil(){
   TNode *bantu;
   bantu = head;
   if(isEmpty()==0){
         do{
               printf("%i ",Bantu->data);
               bantu=bantu->next;
         }while(bantu!=head);
         printf("\n");
   } else printf("masih Kosong");cout<<"Masih kosong\n";
}
```

- Hapus di depan

```
void hapusDepan (){
   TNode *hapus,*bantu;
   int d;
   if (isEmpty()==0){
         if(head->next != head){
               hapus = head;
               d = hapus->data;
               bantu = head->prev;
               head = head->next;
               bantu->next = head;
               head->prev = bantu;
               delete hapus;
         } else {
               d = head->data;
```

```
            head = NULL;
        }
        printf("%i terhapus",d);
    } else printf("Masih kosong\n");
}
```

- Hapus di belakang

```
void hapusBelakang(){
    TNode *hapus,*bantu;
    int d;
    if (isEmpty()==0){
        if(head->next != head){
            bantu = head;
            while(bantu->next->next != head){
                bantu = bantu->next;
            }
            hapus = bantu->next;
            d = hapus->data;
            bantu->next = head;
            delete hapus;
        } else {
            d = head->data;
            head = NULL;
        }
        printf("%i terhapus\n",d);
    } else printf("Masih Kosong");
}
```

latihan :

- Buatlah ilustrasi dari masing-masing potongan program.
- Buat program lengkap dari potongan-potongan program yang ada diatas! Buat agar menjadi seperti menu.
- Buat program untuk memasukkan node baru tetapi diantara node yang sudah ada. Tentukan node yang baru akan berada pada antrian keberapa.

2. Dengan Head dan tail
   - Menggunakan 2 pointer, head dan tail.
   - Head selalu menunjuk node pertama dan tail selalu menunjuk node terakhir

Sebelumnya kita harus mendeklarasikan dulu pointer head :

```
TNode *head, *tail;
```

Setelah kita mendeklarasikan pointer head, kita belum bisa secara langsung mendeklarasikan node yang dituju. Sehingga pointer head harus dibuat bernilai *null* terlebih dahulu :

```
head = NULL;

tail = NULL;
```

untuk mengetahui apakah suatu Linked List kosong atau tidak, kita dapat mengetahuinya dengan mengecek nilai dari pointer Tail-nya.

```
int isEmpty() {
      if(tail==NULL) return 1;
      else return 0;
}
```

Contoh program :

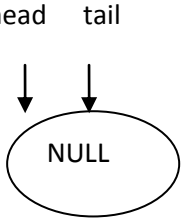
- Penambahan di depan

```
void tambahdata (int databaru){
   TNode *baru;
   baru = new TNode;
   baru -> data = databaru;
   baru -> next = NULL;
   baru -> prev = NULL;
```

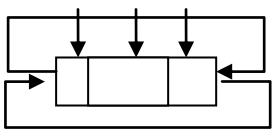
```
   if (isEmpty()==1) {
```

```
            head=baru;
            tail=head;
            head->next=head;
            head->prev=head;
            tail->next=tail;
            tail->prev=tail;
      }
      else {
            baru->next=head;
            head->prev=baru;
            head=baru;
            head->prev=tail;
            tail->next=head;
      }
      printf("data masuk");
   }
```

Penggambaran :

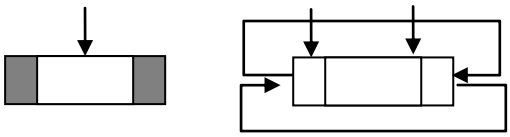



Setelah dibuat node baru dan jika diketahui tail==NULL :

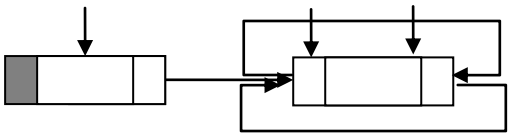



Bila kita membuat node baru lagi maka :

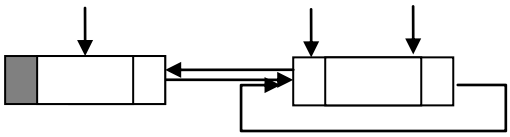
baru head bantu



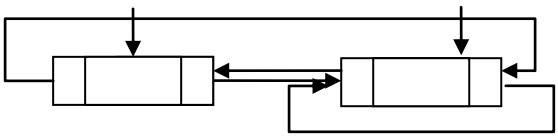
baru head tail



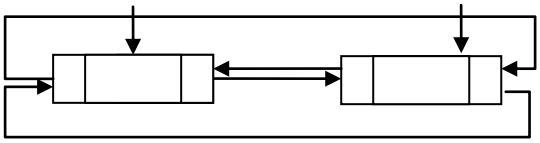
baru head tail



head bantu



head bantu



- Penambahan di belakang

```
void insertBelakang(int databaru){
   TNode *baru;
   baru = new TNode;
   baru->data = databaru;
   baru->next = baru;
   baru->prev = baru;
   if(isEmpty()==1){
         head=baru;
         tail=baru;
```

```
            head->next = head;
            head->prev = head;
            tail->next = tail;
            tail->prev = tail;
      }
      else {
            tail->next = baru;
            baru->prev = tail;
            tail = baru;
            tail->next = head;
            head->prev = tail;
      }
      printf("Data masuk\n");
}
```

- Tampil

```
void tampil(){
      TNode *bantu;
      bantu = head;
      if(isEmpty()==0){
            do{
                  cout<<bantu->data<<" ";
                  bantu=bantu->next;
            }while(bantu!=tail->next);
            printf("\n");
      } else printf("masih kosong\n");
}
```

- Hapus di depan

```
void hapusDepan(){
   TNode *hapus;
   int d;
   if (isEmpty()==0){
         if(head != tail){
               hapus = head;
               d = hapus->data;
               head = head->next;
               tail->next = head;
               head->prev = tail;
               delete hapus;
         } else {
               d = head->data;
               head = NULL;
               tail = NULL;
         }
         printf("%i terhapus\n",d);
   } else printf("Masih Kosong");
}
```

- Hapus di belakang

```
void hapusBelakang(){
   TNode *hapus;
   int d;
   if (isEmpty()==0){
         if(head != tail){
               hapus = tail;
```

```
                d = hapus->data;
                tail = tail->prev;
                tail->next = head;
                head->prev = tail;
                delete hapus;
          } else {
                d = head->data;
                head = NULL;
                tail = NULL;
          }
          cout<<d<<" terhapus\n";
    } else cout<<"Masih kosong\n";
}
```

latihan :

- Buatlah ilustrasi dari masing-masing potongan program.
- Buat program lengkap dari potongan-potongan program yang ada diatas! Buat agar menjadi seperti menu.
- Buat program untuk memasukkan node baru tetapi diantara node yang sudah ada. Tentukan node yang baru akan berada pada antrian keberapa.

## Soal Double Linked List Circular

## Soal Guided

1. Marilah kita buat sebuah Double Linked List Circular(DLLC) sederhana…
   Buatlah sebuah program dengan menggunakan DLLC yang dapat menerima inputan dari belakang maupun dari depan.

```
#include "stdio.h"

typedef struct Tnode
{
 int value;
 Tnode *next;
 Tnode *back;
};

Tnode *baru, *bantu,*head,*tail;
```

```
void tambah(int value)
{
     baru = new Tnode;
     baru->next = baru;
     baru->back = baru;
     baru->value = value;
}
void tambahbelakang(int value){
     tambah(value);
     if(head == NULL)
        head = tail = baru;
     else{
        tail->next = baru;
        baru->back = tail;
        tail = baru;
     }
     tail->next = head;
     head->back = tail;
}
void tambahdepan(int value){
     tambah(value);
     if(head == NULL)
        head = tail = baru;
     else {
        baru->next = head;
        head->back = baru;
        head = baru;
     }
     tail->next = head;
     head->back = tail;
}

void cetak(){
     bantu = head;
     do
     {
      printf("%d",bantu->value);
      bantu = bantu->next;
     }while(bantu!=head);
 }
void menu(){
     int pil;
     int isi;
     do {
         printf("Menu :\n1.Masuk dari depan\n2.Masuk dari
belakang\n3.Cetak\n4.Exit\nMasukkan pilihan anda");
         scanf("%d",&pil);
         switch(pil){
                    case 1 :
                         printf("Masukkan nilai : "); scanf("%d",&isi);
                         tambahdepan(isi);
                         break;
                    case 2 :
                         printf("Masukkan nilai : "); scanf("%d",&isi);
                         tambahbelakang(isi);
                         break;
                    case 3 : cetak();
                         break;
                    case 4 : printf("Terima kasih");
                         break;
```

```
                    default:
                            printf("Tidak ada pilihan tersebut, masukkan
angka dari 1 sampai 3");
          }
     }while(pil!=4);
 }

int main(){
      menu();
      return 0;
}
```

2. Setelah dapat menambahkan dari depan dan belakang, mari kita berlatih untuk dapat menghapus data yang telah di masukkan. Berikut ini adalah contoh program untuk mendelete data dari belakang.

```
#include "stdio.h"

typedef struct Tnode

{

 int value;

 Tnode *next;

 Tnode *back;

};


Tnode *baru, *bantu,*head,*tail, *bantudelete;


void tambah(int value)

{

      baru = new Tnode;

      baru->next = baru;

      baru->back = baru;

      baru->value = value;

}

void tambahbelakang(int value)

{

      tambah(value);
```

```
     if(head == NULL)
        head = tail = baru;
     else
     {
        tail->next = baru;
        baru->back = tail;
        tail = baru;
     }
     tail->next = head;
     head->back = tail;
}
void deletebelakang()
{
     if(head == NULL)
        head = tail = NULL;
     else
     {
        bantudelete = tail;
        tail = tail->back;
        tail->next = head;
        head->back = tail;
        delete bantudelete;
     }
}

void cetak()
{
     if(head != NULL)
     {bantu = head;
     do
     {
```

```
      printf("%d",bantu->value);
      bantu = bantu->next;
     }while(bantu!=head);
     }else
     printf("tidak ada data");
 }
void menu()
{
     int pil;
     int isi;
     do
     {
         printf("Menu :\n1.Masuk dari belakang\n2.Delete dari
belakang\n3.Cetak\n4.Exit\nMasukkan pilihan anda");
         scanf("%d",&pil);
         switch(pil)
         {
                     case 1 :
                          printf("Masukkan nilai : ");
scanf("%d",&isi);
                          tambahbelakang(isi);
                          break;
                     case 2 :
                          deletebelakang();
                          break;
                     case 3 : cetak();
                          break;
                     case 4 : printf("Terima kasih");
                          break;

                     default:
```

```
                                printf("Tidak ada pilihan
tersebut, masukkan angka dari 1 sampai 3");

          }

     }while(pil!=4);

 }


int main(){

      menu();

      return 0;

}
```

**Soal Unguided**

1. Setelah tadi diberi contoh mengenai tambah depan dan belakang, coba sekarang buatlah yang menambah data nya dari tengah☺. Penambahan dilakukan setelah data ke n(inputan user)
2. Setelah dapat menambahkan data setelah data ke n, bagaimana dengan menghapus data ke n? silahkan dicoba dikerjakan…☺

**Soal Take Home**

1. Marilah kita membuat sebuah game☺
   Sebuah mesin roulette memiliki 16 point yang masing-masing memiliki warna dan sebuah nilai angka. Nilai angka yang ada di point-point yang ada adalah angka 1 sampai 16 yang peletakannya secara urut dan warna untuk setiap point berbeda-beda dengan urutan warna merah, putih, dan hitam.

   Tentukan sebuah bola (sebuah pointer random) yang menunjuk ke mesin roulette tersebut dengan perhitungan random tentu saja…

   User bisa memilih akan melempar bola tersebut ke arah kanan atau kearah kiri tergantung pilihan user

   Sebelum memulai user dapat menentukan pilihan dengan cara memilih antara angka 1 – 16 atau warna antara putih, hitam, dan merah. Kemudian memasukkan banyaknya bet(banyaknya point yang ditaruhkan)(pada awal user diberi 1000point)

   Jika tebakan benar maka point user bertambah yaitu 2 kali bet. Jika salah maka point user berkurang sebanyak bet yang dimasukkan

2. Sebuah bank di Indonesia ingin membuat sebuah program dengan menggunakan urutan prioritas. Tujuan program ini adalah untuk memberikan no urut kepada para

nasabahnya. No urut yang diberikan kemudian akan dipanggil oleh program sehingga nasabah yang memiliki no tersebut dapat menuju ke teller yang telah ditunjuk.
Permasalahannya di bank tersebut terdapat sebuah sistem prioritas yang mendahulukan orang2 yang memiliki prioritas, sehingga ketika nasabah yang memiliki prioritas bisa dipanggil terlebih dahulu dari pada nasabah yang tidak memiliki prioritas. Buatlah program menggunakan DLLC untuk mengatasi hal ini…