Praktikum 2

Senarai Berantai (Linked List)

POKOK BAHASAN:

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

TUJUAN BELAJAR:

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami konsep linked list dan mengerti kegunaannya
- ✓ Mengimplementasikan struktur linked list dalam pemrograman
- ✓ Mengidentifikasi permasalahan-permasalahan pemrograman yang harus diselesaikan dengan menggunakan linked list dan menyelesaikannya.

DASAR TEORI:

Secara konseptual, linked list merupakan deretan elemen yang berdampingan. Akan tetapi, karena elemen-elemen tersebut dialokasikan secara dinamis (menggunakan malloc), sangat penting untuk diingat bahwa kenyataannya, linked list akan terpencar-pencar di memori. Pointer dari elemen ke elemen berarti sebagai penjamin bahwa semua elemen dapat diakses.

Single linked list atau biasa disebut linked list terdiri dari elemen-elemen individu, dimana masing-masing dihubungkan dengan pointer tunggal. Masing-masing elemen terdiri dari dua bagian, yaitu sebuah data dan sebuah pointer yang disebut dengan pointer *next*. Dengan menggunakan struktur *two-member* seperti ini, linked list dibentuk dengan cara menunjuk pointer *next* suatu elemen ke elemen yang mengikutinya. Pointer *next* pada elemen terakhir merupakan NULL, yang menunjukkan

akhir dari suatu list. Elemen pada awal suatu list disebut *head*, dan elemen terakhir dari suatu list disebut *tail*.

1. ALOKASI SIMPUL

Ketika sebuah variabel dideklarasikan, terlebih dahulu harus diinisialisasi. Demikian juga dengan pengalokasian secara dinamis. Sehingga, fungsi untuk mengalokasikan sebuah node baru, fungsi allocate_node() menggunakan malloc() untuk mendapatkan memori aktual, yang akan menginisialisasi suatu field data. next selalu diinisialisasi sebagai NULL.

Untuk melihat kemungkinan alokasi memori gagal, maka fungsi allocate_node menghasilkan 0, bila berhasil maka menghasilkan 1. Untuk membebaskan node digunakan fungsi free_node. Fungsi dari alokasi node adalah sebagai berikut :

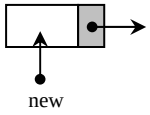
```
int allocate_node(int data, list *new)
{
      new = (list) malloc (sizeof(node));
      if(new==NULL)
            return 0;

      new->datalist = data;
      new->next=NULL;
      return 1;
}
```

Untuk inisialisasi list setelah alokasi untuk node pertama maka ditambahkan statement sebagai berikut:

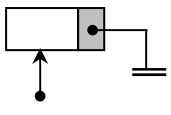
```
head = new;
```

Ilustrasi dari fungsi `allocate_node()` adalah sebagai berikut

```
new = (list) malloc (sizeof(node));
new->datalist = data;
```

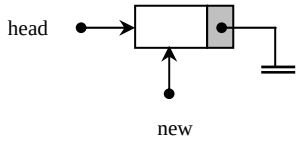



```
new->next=NULL;
```



Untuk inisialisasi list setelah alokasi untuk node pertama ilustrasinya adalah sebagai berikut:

```
head = new;
```

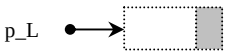


Fungsi `free_node()` menggunakan memori dinamis dengan fungsi free(). free() akan menghancurkan node yang menunjuk ke pointer yang dilewati, sehingga tempat yang ada dapat dipakai untuk lainnya. Akan tetapi, pointer yang melewati free() tidak otomatis menjadi null, tetapi akan menunjuk ke node yang sudah tidak ada. Karena itu pointer harus didefinisikan dengan NULL.

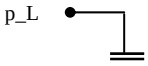
```
void free_node(list p_L)
{
      free(p_L);
      p_L=NULL;
}
```

Ilustrasi dari fungsi `free_node()` dapat dilihat pada gambar berikut :

```
free(*p_L);
```




```
*p_L=NULL;
```



2. OPERASI PADA LINKED LIST

Operasi yang terpenting pada linked list adalah menambahkan node (*insert*) dan menghapus node (*delete*).

2.1 INSERT

Fungsi insert pada linked list meliputi :

- insert sebagai node awal (*head*) dari linked list
- insert setelah node tertentu
- insert sebelum node tertentu
- insert sebagai node akhir (*tail*) dari linked list

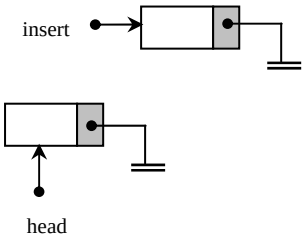
Insert sebagai node awal (*head*) dari linked list

Statement kelanjutan dengan deklarasi seperti diatas untuk insert sebagai node awal (head) dari linked list adalah sebagai berikut:

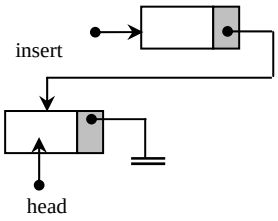
```
void insertashead(list insert)
{
      insert->next=head;
      head = insert;
}
```

ilustrasi dari fungsi diatas adalah sebagai berikut:

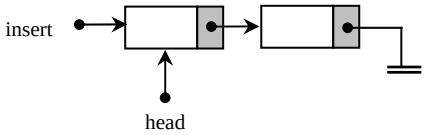
kondisi awal




```
insert->next=head;
```




```
head = insert;
```



Insert setelah node tertentu

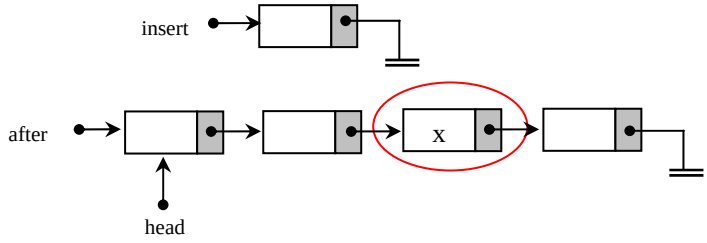
Statement untuk insert setelah node tertentu dari linked list adalah sebagai berikut:

```
void insertafternode(int x, list insert)
{
      list after;
      after = head;
      do
            after = after->next;
```

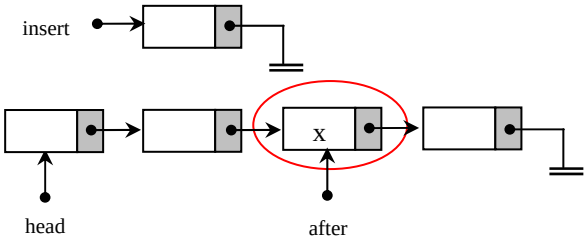
```
        while (after->datalist != x);
        insert->next = after->next;
        after->next = insert;
}
```

ilustrasi dari fungsi diatas adalah sebagai berikut:

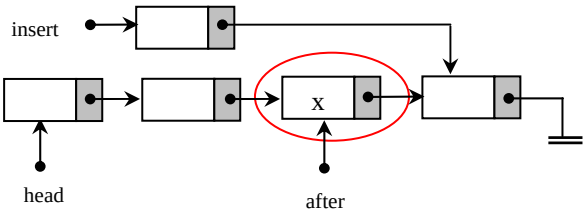
```
after = head;
```



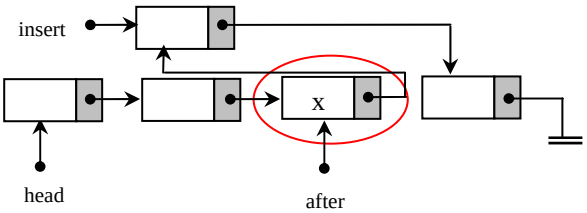

```
do
        after = after->next;
while (after->datalist != x);
```




```
insert->next = after->next;
```




```
after->next = insert;
```



Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Pointer `next` dari elemen baru menunjuk elemen setelah elemen tertentu
2. Pointer `next` elemen sebelumnya menunjuk ke elemen baru

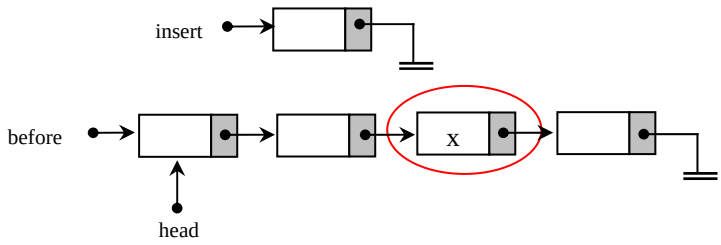
Insert sebelum node tertentu

Statement untuk *insert* setelah node tertentu dari linked list adalah sebagai berikut:

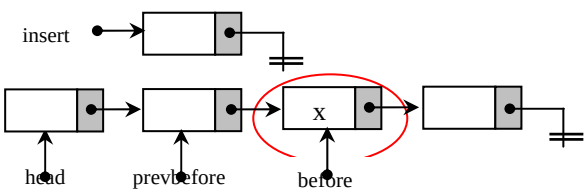
```
void insertbeforenode(int x, list insert)
{
      list before, prevbefore;
      if (head->datalist = x)
            insertashead(insert)
      else
      {
            before = head;
            do
                  prevbefore = before;
                  before = before->next;
            while (before->datalist != x);
            insert->next = before;
            prevbefore->next = insert;
      }
}
```

ilustrasi dari fungsi diatas adalah sebagai berikut:

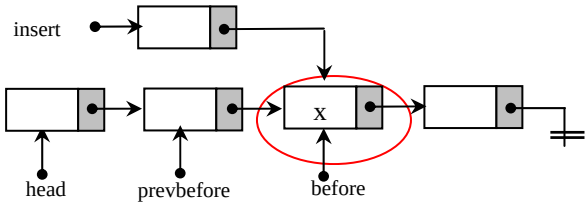
```
before = head;
```



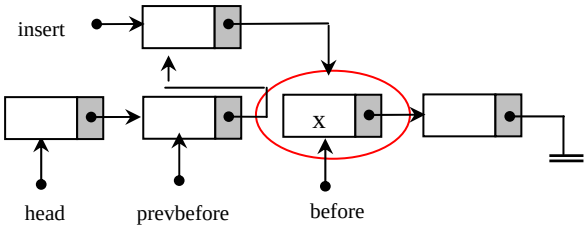

```
do
      prevbefore = before;
      before = before->next;
while (before->datalist != x);
```



```
insert->next = before;
```




```
prevbefore->next = insert;
```



Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Telusuri list sampai elemen tertentu, catat juga elemen sebelumnya
2. Lakukan penyisipan sebelum elemen tertentu tersebut

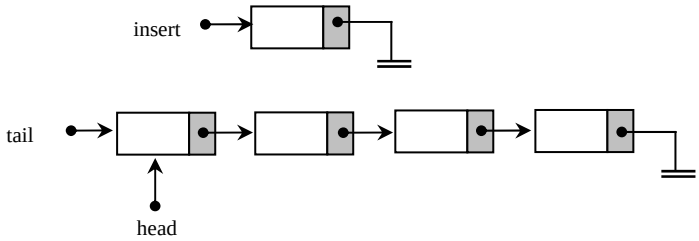
Insert di akhir (*tail*) dari linked list

Statement untuk *insert* di akhir dari list adalah sebagai berikut:

```
void insertastail(list insert)
{
      list tail;
      tail = head;
      do
            tail = tail->next;
      while (tail->next != NULL);
      tail->next = insert;
      tail = tail->next;
}
```

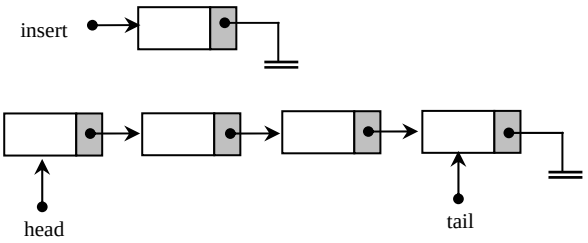
ilustrasi dari fungsi diatas adalah sebagai berikut:

```
tail = head;
```

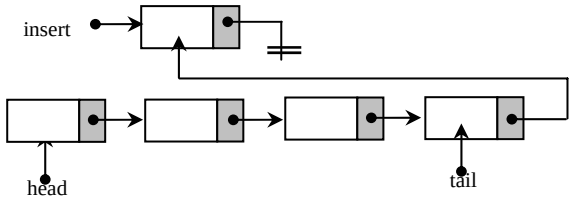



```
do
```

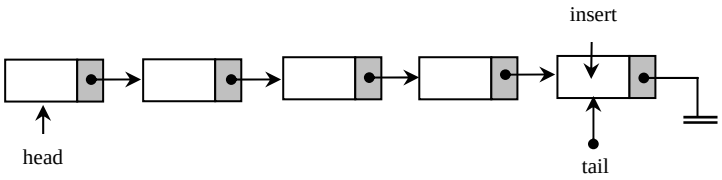
```
      tail = tail->next;
while (tail->next != NULL);
```




```
tail->next = insert;
```




```
tail = tail->next;
```



Langkah-langkah untuk proses di atas adalah sebagai berikut:

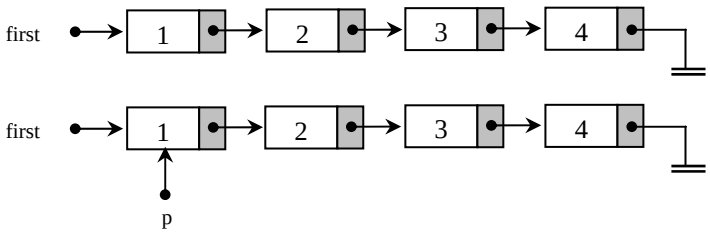
1. Telusuri list sampai elemen terakhir (`tail->next=NULL`)
2. Lakukan pentisipan setelah elemen terakhir

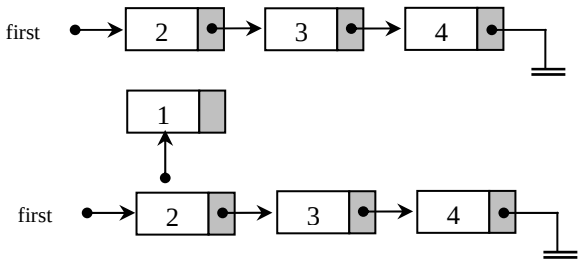
2.2 DELETE

Fungsi *delete* pada linked list meliputi :

- *delete* sebagai simpul pertama(*head*) dari linked list
- *delete* setelah simpul tertentu
- *delete* simpul terakhir

Penghapusan Simpul Pertama:

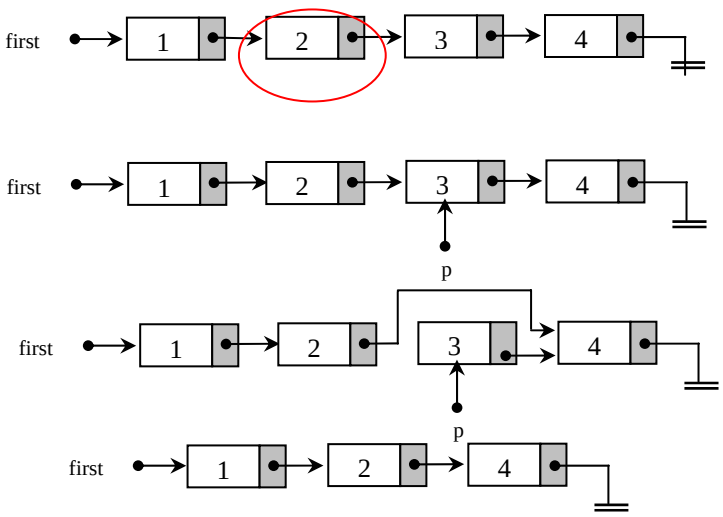




Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Pointer `first` diarahkan pada data ke-2
2. Pointer `p` diarahkan pada data ke-1
3. Bebaskan pointer `p` (secara otomatis data ke-1 terhapus)

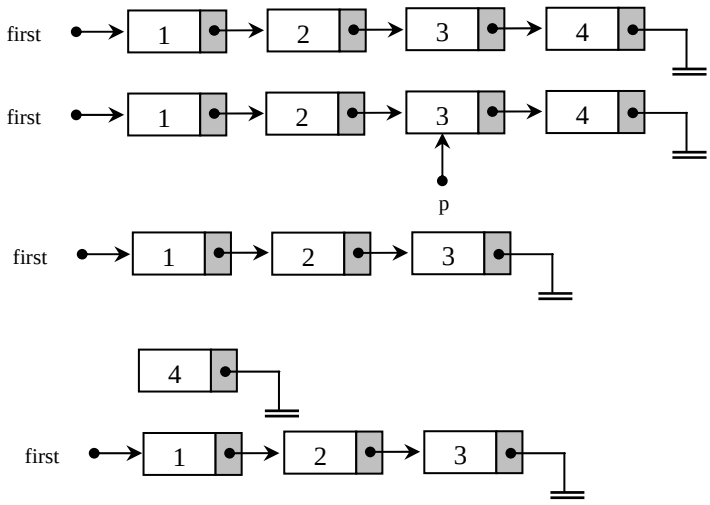
Penghapusan Setelah Simpul Tertentu



Langkah-langkah untuk proses di atas adalah sebagai berikut:

1. Arahkan pointer `first` s/d data yang ditunjuk
2. Pointer `p` diarahkan pada `first->next`
3. Arahkan pointer `first->next` pada `p->next`
4. Bebaskan pointer `p` (secara otomatis data setelah simpul tertentu terhapus)

Penghapusan Simpul Terakhir



Langkah-langkah untuk proses di atas adalah sebagai berikut:

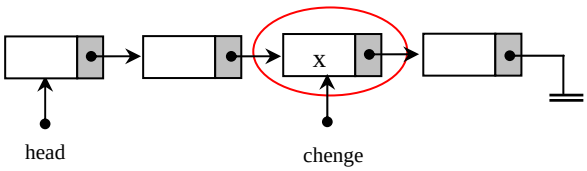
1. Telusuri simpul s/d `first->next = NULL`
2. Arahkan pointer `p` ke `first`
3. Bebaskan pointer `p->next` (Simpul Terakhir)
4. Arahkan `p->next` ke `NULL`

2.3 MERUBAH ISI DARI SUATU SIMPUL PADA LINKED LIST

Untuk merubah isi dari suatu simpul pada linked list, maka yang harus dilakukan adalah:

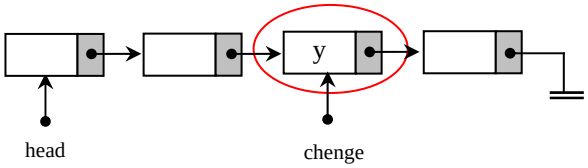
1. Menulusuri linked list sampai didapatkan simpul yang akan dirubah

```
do
   change = change->next;
while (change->datalist != x);
```



2. Merubah isi dari simpul yang sudah didapatkan

```
change->datalist != x
```



TUGAS PENDAHULUAN:

1. Buatlah flowchart untuk tiap operasi pada linked list untuk Membuat Linked List sebagai Simpul Baru
2. Buatlah flowchart untuk menyisipkan Simpul setelah Simpul Tertentu
3. Buatlah flowchart untuk menyisipkan Simpul sebelum Simpul Tertentu
4. Buatlah flowchart untuk menyisipkan Simpul sebagai Simpul Terakhir
5. Buatlah flowchart untuk menghapus Simpul, baik Simpul Pertama, Simpul di Tengah ataupun Simpul di Akhir

PERCOBAAN:

1. Buatlah workspace untuk praktikum Struktur Data dengan menggunakan Visual C++.
2. Buatlah project untuk praktikum kedua
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 Menambahkan Simpul Baru sebagai LIFO

```
struct dtnilai
{
      char nrp[10];
      char nama[20];
      double nilai;
      struct dtnilai *next;
};

struct dtnilai *ujung;

void sisip_awal_LIFO()
{
      struct dtnilai *tampung;
      int j=0;char jawab[2];
      while(1)
      {
            ujung=(struct dtnilai*)malloc(sizeof(struct dtnilai));
            fflush(stdin);
            printf("NRP :");scanf("%s",&ujung->nrp);
            printf("Nama      :");scanf("%s",&ujung->nama);
            printf("Nilai Test :");scanf("%lf",&ujung->nilai);
            if(j==0)
            {
                  ujung->next=NULL;
                  tampung=ujung;
            }
            else
            {
                  ujung->next=tampung;
                  tampung=ujung;
            }
            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 : Menambah Simpul Baru sebagai FIFO

```
void sisip_awal_FIFO()
{
      struct dtnilai *tampung, *awal;
      int j=0;char jawab[2];
```

```
      while(1)
      {
            awal=(struct dtnilai*)malloc(sizeof(struct dtnilai));
            fflush(stdin);
            printf("NRP :");scanf("%s",&awal->nrp);
            printf("Nama      :" );scanf("%s",&awal->nama);
            printf("Nilai Test :" );scanf("%lf",&awal->nilai);
            awal->next=NULL;
            if(j==0)
            {
                  ujung=awal;
                  tampung=ujung;
            }
            else
            {
                  tampung->next=awal;
                  tampung=tampung->next;
            }
            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 3 : Fungsi untuk Menyisipkan Simpul Setelah Simpul Tertentu

```
void sisip_stl_simpul()
{
      struct dtnilai *sisip; struct dtnilai *stl;
      char cari[20];

      if (ujung == NULL)
            pritnf("List Belum Terbentuk. Buatlah Dulu!");
      else
      {
            sisip=(struct dtnilai*)malloc(sizeof(struct dtnilai));
            printf("Masukkan Data Yang Akan Disisipkan \n");
            printf("NRP :"); gets(sisip->nrp);
            printf("Nama      :"); gets(sisip->nama);
            printf("Nilai Test :"); gets(sisip->nilai);

            printf("Data disisipkan setelah data ? (nama) : ";
            gets(cari);

            stl = ujung;
```

```
            while(strcmp(stl->nama,cari)!=0)
            {
                  stl=stl->next;
            }

            sisip->next = stl->next;
            stl->next = sisip;
      }
}
```

Percobaan 4 : Fungsi untuk Menampilkan Isi dari List

```
void tampil_list()
{
      struct dtnilai *tampil;

      printf("Data Mahasiswa yang telah diinputkan :\n");
      printf("NRP\tNama\tNilai\n");

      tampil = ujung;

      while(tampil!=NULL)
      {
            printf("%s\t%s\t%6.2f\n",tampil->nrp,tampil->nama,
tampil->nilai);
            tampil=tampil->next;
      }
}
```

Percobaan 5 : Fungsi untuk Menghapus Simpul Tertentu

```
void hapus_simpul_tertentu()
{
      struct dtnilai *hapus; struct dtnilai *sblhapus;
      char cari[20];

      if (ujung == NULL)
            printf("List Belum Terbentuk. Buatlah Dulu!");
      else
      {
            printf("Data yang akan dihapus? (nama) : ");
            gets(cari);

            hapus = ujung;

            if(strcmp(hapus->nama,cari)==0)
            {
                  ujung = ujung->next;
                  free(hapus);
            }
```

```
            else
            {
                  while(strcmp(hapus->nama,cari)!=0)
                  {
                        sblhapus=hapus;
                        hapus=hapus->next;
                  }

      if (hapus->next == NULL)
                  {
                        sblhapus->next=NULL;
                        free(hapus);
                  }else{
                        sblhapus->next = hapus->next;
                        free(hapus);
                  }
            }
      }
}
```

Percobaan 6 : Fungsi untuk Merubah Data pada Simpul Tertentu

```
void rubah_data()
{
      struct dtnilai *rubah;
      char cari[20];
      char nrpbaru[10];
      char namabaru[20];
      double nilaibaru;

      rubah = ujung;

      fflush(stdin);
      printf("Masukkan nama dari data yang akan dirubah : ");
      scanf("%s",&cari);

      while (strcmp(rubah->nama,cari)!=0)
      {
            rubah = rubah->next;
      }

      printf("Masukkan data yang akan dirubah.\n");
      printf("NRP :");scanf("%s",&nrpbaru);
      printf("Nama      :" );scanf("%s",&namabaru);
      printf("Nilai Test :" );scanf("%lf",&nilaibaru);

      strcpy(rubah->nrp,nrpbaru);
      strcpy(rubah->nama,namabaru);
      rubah->nilai=nilaibaru;
}
```

LATIHAN:

1. Masalah aritmatika polinom adalah membuat sekumpulan subrutin manipulasi terhadap polinom simbolis (symbolic Polynomial). Misalnya: P1 = $6x^8 + 8x^7 + 5x^5 + x^3 + 15$

   P2 = $3x^9 + 4x^7 + 3x^4 + 2x^3 + 2x^2 + 10$

   P3 = $x^2 + 5$

   Representasikan bilangan polinom dengan menggunakan linked list dan buatlah prosedur-prosedur untuk :

   - Menyisipkan simpul di awal jika pangkat yang dimasukkan lebih dari pangkat tertinggi dari bilangan polinomial.
   - Menyisipkan simpul di tengah jika pangkat dari bilangan yang kita sisipkan berada di tengah.
   - Menyisipkan simpul di akhir jika pangkat dari bilangan yang disisipkan adalah 0.
   - Menghapus simpul, baik di awal, di tengah, ataupun di akhir.

2. Implementasikan sebuah single linked list yang merepresentasikan data mahasiswa. Data mahasiswa berupa nrp, nama, alamat, indeks prestasi. Buatlah fungsi-fungsi untuk membangun single linked list, menelusuri, menambah simpul, menghapus simpul.

3. Tambahkan tampilan di output setelah anda mengerjakan percobaan di atas dengan penghitungan indeks prestasi rata-rata, dimana
   iprata=total/jumlah_siswa;
   total didapatkan dari menambahkan IP yang didapat tiap mahasiswa.