MODUL PRAKTIKUM
STRUKTUR DATA

Tim Dosen Teknik Informatika
dan Sistem Informasi

FAKULTAS ILMU KOMPUTER
UNIVERSITAS ESA UNGGUL

ARRAY

Pertemuan kali ini kita akan kembali membahas materi yang sudah diberikan di algoritma dan pemrograman, yaitu array. Secara singkat, array adalah suatu tipe data terstruktur yang berupa sejumlah data sejenis (bertipe data sama) yang jumlahnya tetap dan diberi suatu nama tertentu.

Array dapat berupa array 1 dimensi, 2 dimensi, bahkan n-dimensi.

DEKLARASI

```
tipe_data nama_var_array [ukuran];
```

tipe_data	: menyatakan jenis tipe data elemen larik (int, char, float, dll)
nama_var_array	: menyatakan nama variabel yang dipakai.
ukuran	: menunjukkan jumlah maksimal elemen larik.

```
Contoh :
```

Int nilai[6];

INISIALISASI

Menginisialisasi array sama dengan memberikan nilai awal array pada saat didefinisikan.

```
int nilai[6] = {8,7,5,6,4,3};
```

Contoh diatas berarti berarti anda memesan tempat di memori komputer sebanyak 6 tempat

dengan indeks dari 0-5, dimana indeks ke-0 bernilai 8, ke-1 bernilai 7, dst, dan dimana semua elemennya bertipe data integer.

PENGAKSESAN

```

```

```
nama_var_array [indeks];
```

Pengisian dan pengambilan nilai pada indeks tertentu dapat dilakukan dengan mengeset nilai atau menampilkan nilai pada indeks yang dimaksud. Pengaksesan elemen array dapat dilakukan berurutan atau random berdasarkan indeks tertentu secara langsung.

Contoh pengisian langsung saat deklarasi:

```
#include <stdio.h>


void main ()
{ int billy [] = {16, 2, 77, 40, 12071};
  int n, result=0;
  for ( n=0 ; n<5 ; n++ )
  {
    result += billy[n];
  }
  printf("%d",result);
}
```

Contoh pengaksesan dan pengisian langsung ke tiap elemen dari array:

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


void main ()
{
int A [5]={20,9,1986,200,13},n,edit;
  clrscr();
  printf("Data yang lama\n");
  for (n=0;n<5;n++)
  {
    printf("%i ",A[n]);
  }
```

```
  printf("\nData yang baru : \n");
  A[0]=4;
  A[1]=2;
  A[2]=1;
  A[3]=3;
  A[4]=5;
  for (n=0;n<5;n++)
  {
    printf("%i ",A[n]);
  }
}
```

Contoh penghapusan data(elemen) pada array:

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

void main ()
{ int A [5]={20,9,1986,200,13},n,hapus;
  clrscr();
  printf("Data yang lama\n");
  for (n=0;n<5;n++)
  {
    printf("%i ",A[n]);
  }
  printf("data yang ingin dihapus : ");
  scanf("%i",&hapus);
  printf("\nData yang baru : \n");
  for (n=hapus-1;n<5-1;n++)
  {
    A[n]=A[n+1];
  }
  for (n=0;n<4;n++)
```

```
  {
    printf("%i ",A[n]);
  }
}
```

LATIHAN TERBIMBING ARRAY

1. Penghapusan data pada array.
   Ketentuan : data yang akan dihapus adalah data pada indeks ke-0 (data paling depan).

   Source code :

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

void main(){
 int data[]={1,6,2,9,12,87,43,11};
 int n=8;

  do{
       //menampilkan data
       for(int i=0;i<n;i++)
            printf("%i ",data[i]);

       getch();
       printf("\n");
```

Cobalah program di atas.

Apakah yang terjadi?

Bagaimana jika penghapusan terjadi pada data yang berada di indeks terakhir? Bagian program yang mana yang harus diubah?

STRUCT

- Bentuk struktur data yang dapat menyimpan variabel-variabel dalam 1 nama, namun memiliki tipe data yang berbeda ataupun sama. Variable-variabel tersebut memiliki kaitan satu sama yang lain.

Bentuk umum :

```
typedef struct nama_struct{
tipe_data <nama_var>;
tipe_data <nama_var>;
....
};
```

DEKLARASI

Ada 2 cara pendeklarasian struct, yaitu :

Deklarasi 1:

```
typedef struct Mahasiswa {
char NIM[8];
char nama[50];
float ipk;
};
```

Deklarasi 2 :

```
struct {
char NIM[8];
char nama[50];
float ipk;
} mhs;
```

Contoh struct:

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

void main()

{
 struct orang
 {
      char nama[40];
      short umur;
 }saya;
 printf("nama : ");
 cin.getline(saya.nama,40);
 printf("umur :" );
 scanf("%i",&saya.umur);
 printf("%s berumur %i",saya.nama,saya.umur);
}
```

ARRAY OF STRUCT

Apabila hendak menggunakan 1 struct untuk beberapa kali, ada 2 cara :

1. Deklarasi manual

Contoh :

```
   #include <stdio.h>
      typedef struct Mahasiswa {
         char NIM[8];
      char nama[50];
         float ipk;
      };
void main()
{
         Mahasiswa a,b,c;
```

```
    ……
          ……
          ……
}
```

artinya struct mahasiswa digunakan untuk 3 variabel, yaitu a,b,c

2. Array of struct

Contoh :

```
   #include <stdio.h>
      typedef struct Mahasiswa {
         char NIM[8];
      char nama[50];
         float ipk;
      };
void main()
{
         Mahasiswa mhs[3];
   ……
          ……
          ……
}
```

artinya struct mahasiswa dapat digunakan untuk tiga variabel mhs, yaitu mhs[0], mhs[1], dan mhs[2].

Contoh lainnya :

```
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
typedef struct orang
{
```

```
 char nama[30];
 short umur;
 };
void main()

{
 orang saya[5];
 int i,x;
 for(i=0;i<=4;i++)
 {
printf("nama ke-%i : ",i+1);
cin.getline(saya[i].nama,30);
printf("umur ke-%i : ",i+1);
scanf("%i",saya[i].umur);
printf("%s berumur %i",saya[i].nama,saya[i].umur);
 }
 for(x=0;x<=4;x++)
 {
printf("nama %s berumur %d",saya[x].nama,saya[x].umur);
 }
 }
```

LATIHAN TERBIMBING STRUCT

Buatlah struct untuk buku dengan deklarasi manual.

Ketentuan :

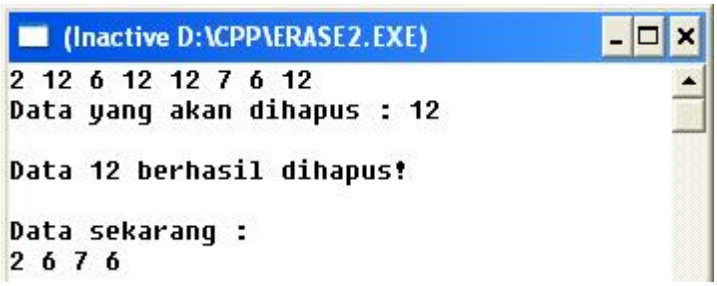
Yang harus disimpan adalah judul buku, tahun terbit dan harga buku.

Source code :

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

typedef struct buku{
      char judul[15];
      int tahun_terbit;
      int harga;
};

void main(){

 buku book;
```

LATIHAN - LATIHAN MANDIRI DI KELAS

1. Program penghapusan data dengan inputan berupa angka yang ingin dihapus oleh user.
   Ketentuan :
   - Semua data yang sesuai dengan inputan user akan terhapus.
   - Bonus jika terdapat counter untuk menghitung berapa data yang terhapus.
   - Capture :



2. Program untuk melakukan update data.
   Ketentuan :

- Terdapat dua inputan, yaitu inputan data yang akan diubah dan data baru (data pengganti)
- Semua data yang sesuai dengan inputan user akan diupdate nilainya.

3. Program untuk melakukan penambahan data.
   Ketentuan :
   - Penambahan data dapat dilakukan di mana saja.
   - Inputan dari user berupa :
     - Nilai yang akan ditambahkan
     - Indeks ke berapa yang dituju
   - Setelah penambahan, maka jumlah data akan bertambah dan posisi data akan bergeser sesuai dengan penambahan yang telah dilakukan.

4. Buatlah struct untuk data lagu yang berisi tentang judul lagu, penyanyi, tahun produksi, nomor track dan kode album.
   Ketentuan :
   - program ini akan memiliki dua buah struct, yaitu struct lagu dan struct kodeRBT.
   - Jumlah data yang diinputkan dinamis (maks. 20 lagu)

LATIHAN – LATIHAN MANDIRI DI RUMAH / TES

1. Buatlah menu add, edit, view dan delete data menggunakan array.
   Note : operasi-operasi tersebut dapat dilakukan pada data dan indeks mana saja. (Inputan data dan indeks dinamis).
2. Buatlah dengan menggunakan struct dan array 1 dimensi : record peminjaman buku di perpustakaan.
   Data yang akan ditampilkan sebagai output adalah :
   - Nama
   - NIM
   - Tanggal peminjaman (dd/mm/yyyy)
   - Kode buku, dengan format **nomor rak-kategori buku** :
     - Nomor rak (inputan terserah)
     - Kategori : R (Referensi) atau U (Umum)
     - Contoh : 1234-R

   Gunakan tiga struct untuk kasus ini!

Gabungkan soal 1 dan 2 di atas sehingga menghasilkan program berisi struct yang dapat melakukan fungsi add, edit, view dan delete

Contoh tampilan program :

```
Berapa data yang anda inputkan ? 1
Data ke-1:
      Judul lagu : Bersamamu
      Penyanyi : Vierra
      Tahun Produksi : 2009
      Nomor Track : 3
      Kode Album : MFL

Data lagu yang anda miliki :
   Data 1:
      Penyanyi : Vierra
      Judul lagu : Bersamamu
      Kode RBT : 3-MFL
      Tahun : 2009
```

<Asisten-asisten dapat memodifikasi atau mengembangkan soal-soal di atas sesuai dengan kelasnya>

END

Searching Array

Pada semester yang lalu kita sudah mempelajari array, baik array 1 dimensi maupun array 2 dimensi. Pada bab ini, kita akan mempelajari beberapa cara untuk melakukan pencarian suatu nilai dalam sebuah array 1 dimensi (pada array 2 dimensi sama saja).

Teknik pencarian data dari array yang paling mudah adalah dengan cara ***sequential search,*** dimana data dalam array dibaca 1 demi satu, diurutkan dari index terkecil ke index terbesar, maupun sebaliknya.

Contoh :

Array :

int a[5] = {0,3,6,10,1} (index array pada bahasa c dimulai dari index ke 0 !!!)

jika kita ingin mencari bilangan 6 dalam array tersebut, maka proses yang terjadi kita mencari

1. dari array index ke-0, yaitu 0, dicocokan dengan bilangan yang akan dicari, jika tidak sama, maka mencari ke index berikutnya
2. pada array index ke-1, juga bukan bilangan yang dicari, maka kita mencari lagi pada index berikutnya
3. pada array index ke-2, ternyata bilangan yang kita cari ada ditemukan, maka kita keluar dari looping pencarian

contoh source :

```
Algoritma tersebut di atas dapat dipecahkan dengan program sebagai berikut:



#include<iostream.h>
#include<conio.h>

int linier_search(int [],int,int);
main()
```

```
{
clrscr();
int array_size;
array_size = 10;
int array[10] ={25,36,2,48,0,69,14,22,7,19};
cout<<"Isi dari array adalah:"<<endl;
cout<<"\n Array :";
for (int count=0;count<array_size;count++)
{
  cout<<"\n Array ["<<count<<"]"<<array[count]<<endl;
}
int searching_element=0;
int flag = 0;
cout<<"\n \n Masukkan data yang akan Anda cari = ";
cin>>searching_element;

flag = linier_search(array, array_size,searching_element);
if(flag!=-1)
cout<<"\n Data tersebut ditemukan pada posisi array[ "<<flag<<"]";
else
cout<<"\n  Data tersebut tidak ditemukan ";
getch();return 0;
}

int linier_search(int array[], int array_size, int element)
{
int flag = -1;
for (int count=0;count<array_size;count++)
{
if (element==array[count])
{
flag=count;
```

```
break;
}
}
return flag;
}
```

output :

Metode pencarian yang kedua adalah ***binary search,*** pada metode pencarian ini, **data harus diurutkan terlebih dahulu**. Pada metode pencarian ini, data dibagi menjadi dua bagian (secara logika), untuk setiap tahap pencarian.

Algoritma binary search :

1. Data diambil dari posisi 1 sampai posisi akhir N
2. Kemudian cari posisi data tengah dengan rumus: (posisi awal + posisi akhir) / 2
3. Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?
4. Jika lebih besar, maka proses pencarian dicari dengan posisi awal adalah posisi tengah + 1
5. Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir adalah posisi tengah – 1
6. Jika data sama, berarti ketemu.

Contoh source binary search :

```
#include<iostream.h>
#include<conio.h>

int binary_search(int [],int,int);
main()
{
clrscr();
int array_size;
array_size = 10;
int array[10] ={0,6,9,12,20,23,29,32,47,79};
```

```
cout<<"Isi dari array adalah:"<<endl;
for (int count=0;count<array_size;count++)
{
  cout<<"Array ["<<count<<"]"<<array[count]<<endl;
}
int searching_element=0;
int flag = 0;
cout<<"\n \n Masukkan data yang akan Anda cari = ";
cin>>searching_element;

flag = binary_search(array, array_size,searching_element);
if(flag!=-1)
cout<<"\n Data tersebut ditemukan pada posisi array["<<flag<<"]";
else
cout<<"\n  Data tersebut tidak ditemukan ";
getch();return 0;
}

int binary_search(int array[], int array_size, int element)
{
int start= 0;
int end=array_size-1;
int middle;
int position = -1;
middle = (start + end)/2;
do{
if(element<array[middle])
end=middle-1;
else if (element>array[middle])
start=middle+1;
middle=(start+end)/2;
}
```

```
while(start<=end && array[middle]!=element);
if(array[middle]==element)
position=middle;return position;}
```

Interpolation searchmerupakan salah satu metode pencarian yang dapat digunakan. Seperti pada *binary search,* data yang harus diurutkan terlebih dahulu, sebelum dapat dilakukan pencarian dengan metode ini. Pada metode pencarian ini, kita mencoba menebak letak data yang kita cari, dengan perhitungan

$$Posisi = \frac{kunci - data[low]}{data[high] - data[low]} x(high - low) + low$$

- Jika data[posisi] > data yg dicari, high = pos – 1
- Jika data[posisi] < data yg dicari, low = pos + 1

Contoh source code interpolation search :

d:\kuliah\ukdw\asiste-1\strukt-1\source\search-1\interpolation.cpp

```
#include<stdio.h>
#include<math.h>
void main(){
int array_a[10]={11,12,21,25,30,34,45,65,73,81};
int low, high, pos, cari, flag=0;
float posisi;
low=0;
high=9; //dari banyak_data-1
printf("data yang anda cari : ");
scanf("%i",&cari);
do{
   posisi=(float)((cari-array_a[low])/(array_a[high]-array_a[low]))*(high-low)+low;
   pos=floor(posisi);
   if(array_a[pos]==cari){
      flag=1;
      break;
      }
   if(array_a[pos]>cari) high=pos-1;
   else if(array_a[pos]<cari)
   low=pos+1;
   }while(cari>=array_a[low] && cari<=array_a[high]);
   if(flag==1) printf("data ditemukan");
   else printf("data tidak ditemukan");
}
```

Tambahan materi :

break;

digunakan untuk keluar dari suatu blok perintah

continue;

digunakan untuk mem *by-pass* satu iterasi pada perulangan

Latihan 1

1. Buatlah sebuah program yang dapat menerima inputan data kedalam sebuah array.
2. Lanjutan dari nomor 1, gunakan sequensial search untuk mencari sebuah nilai dari array tersebut, dan gantilah nilainya.
3. Coba gunakan metode pencarian lainnya
4. Buatlah menu untuk program tersebut (1. Sequential search, 2. Binary Search, 3. Interpolation Search)

Latihan 2

1. Buatlah sebuah program yang dapat mencari dan menampilkan suatu bilangan yang dicari beserta indexnya

   contoh :

   isi array : 12, 14, 15, 12, 5

   data yang dicari : 12

   output: data 12 ditemukan pada index ke 0 dan 3

   petunjuk :

   - coba tampung index array yang ditemukan pada sebuah array baru
   - cara 2, langsung tampilkan index array jika data ditemukan

2. buatlah program untuk mencari data pada array 2 dimensi (bisa ditambahkan kode program untuk memberi inputan data dan ukuran array)

   contoh :

   data array :

   1 3 2

   10 5 8

   15 24 10

   yang dicari : 24

   output : data 24 berada pada posisi [2][1]

   yang dicari : 2

   output : data 2 berada pada posisi [0][2]

petunjuk : gunakan sequential search, karena data tidak diurutkan, terdapat 2 looping untuk proses pencarian

GUIDED

Untuk menghitung jumlah vokal berapa dan konsonan berapa.

```
// Menghitung jumlah vokal konsonan angka
# include <stdio.h>
# include <ctype.h>
# include <string.h>
void main()
{
   char kata[100],b=0,c=0,d=0;
   gets(kata);
   for (int a=0;kata[a];a++)
   {
      if(toupper(kata[a])== 'A' || toupper(kata[a])=='I'|| toupper(kata[a])=='U'
      || toupper(kata[a])=='E'|| toupper(kata[a])=='O' )
      {
         b++;
      }
      else if (isdigit (kata[a]))
      {
         c++;
      }
      else if (isspace (kata[a])){}
      else
      d++;
   }
   printf("Jumlah vokal=%i\t ",b);
   printf("Jumlah konsonan=%i\t",d);
   printf("Jumlah numerik=%i\t",c);
}
```

UNGUIDED

1. Buat program untuk mencari angka yang diinputka,n banyak data inputan user, lalu cari di indeks keberapa, dan berapa jumlah angkanya kalo ada yang sama. (Array 1 dimensi)

2. Buat program hampir sama seperti soal diatas dan bisa mengetahui jumlah bilangan genap ada berapa,ganjil berapa. (Diminta inputan data dalam array 2 dimensi misal: (2x6))

Take Home

1. Buat program untuk mencari suatu data dan inputan berupa kalimat, hitung konsonan,vokal,numerik.

   Seperti contoh:

```
Input : aku dan aka
Output :
Vokal = 5 = a u a a a
Konsonan = 4 = k d n k
Masukkan data yang akan Anda cari:s

 Data tidak ada...
```

2. Buat searching kata, yang ada dalam kalimat!

Sorting Algorithm

Bubble Sort -- Selection Sort -- Insertion Sort -- Exchange Sort

Sorting Algorithm

Salah satu bagian penting dari struktur data adalah **sorting** atau pengurutan data. Ada banyak sekali Algoritma pengurutan data di dunia komputer, yatu : **bubble sort, selection sort, insertion sort, exchange sort,** quick sort, merge sort, dll.

Setiap algoritma memiliki kelebihan dan kekurangan masing – masing, kita akan mempelajari cukup 4 algoritma saja, yaitu bubble sort, selection sort, insertion sort, dan exchange sort. Bila ingin mempelajari algoritma yang lain silakan berkunjung ke www.wikipedia.org .

Bubble Sort

Perhatikan kode berikut :

d:\bubble.cpp

```
#include "stdio.h"

void main(){

   int  A[5]={3,4,1,2,8},i,j,tampung;

   printf("Sebelum sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }

   for (i=1;i<5;i++){
      for(j=5-1;j>=i;j--){
         if (A[j]<A[j-1]){
            tampung=A[j];
            A[j]=A[j-1];
            A[j-1]=tampung;
         }
      }
   }

   printf("\n\nSetelah sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }
}
```

Task 1

1. Apa yang dilakukan program diatas ?
2. Lakukan untuk pengurutan sebaliknya!
3. Pengurutan diatas dilakukan dari depan atau belakang? Buat program untuk sebaliknya!
4. Buat program agar *user* bisa inputkan data secara dinamis, baik untuk *ascending*, maupun *descending*!

5. Tambahkan kode agar *user* dapat melihat proses pengurutan data!

   Note : Ascending adalah pengurutan data dari terkecil menuju terbesar, sedangkan descending adalah pengurutan dari data terbesar menuju terkecil.

Exchange Sort

Perhatikan kode berikut :

d:\exchange.cpp

```
#include "stdio.h"

void main(){

   int  A[5]={3,4,1,2,8},i,j,tampung;

   printf("Sebelum sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }

   for (i=0;i<5-1;i++){
      for(j=i+1;j<5;j++){
         if (A[i]<A[j]){
            tampung=A[i];
            A[i]=A[j];
            A[j]=tampung;
         }
      }
   }

   printf("\n\nSetelah sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }
}
```

Task 2

1. Apa yang dilakukan program diatas ?
2. Lakukan untuk pengurutan sebaliknya!
3. Buat program agar *user* bisa inputkan data secara dinamis, baik untuk *ascending*, maupun *descending*!

4. Tambahkan kode agar *user* dapat melihat proses pengurutan data!
5. Apa perbedaan antara exchange sort dan bubble sort!

Selection Sort

Perhatikan kode berikut :

d:\selectio.cpp

```
int  A[5]={3,4,1,2,8},i,j,tampung,pos;

printf("Sebelum sorting : \n");
for (i=0;i<5;i++){
   printf("%i ",A[i]);
}

for (i=0;i<5-1;i++){
   pos=i;
   for(j=i+1;j<5;j++){
      if (A[j]<A[pos]){
         pos=j;
      }
   }
   if (pos != i){
      tampung=A[pos];
      A[pos]=A[i];
      A[i]=tampung;
   }
}

printf("\n\nSetelah sorting : \n");
for (i=0;i<5;i++){
   printf("%i ",A[i]);
}
}
```

Task 3

1. Apa yang dilakukan program diatas ?
2. Lakukan untuk pengurutan sebaliknya!

3. Apa fungsi **pos**?
4. Buat program agar *user* bisa inputkan data secara dinamis, baik untuk *ascending*, maupun *descending*!
5. Tambahkan kode agar *user* dapat melihat proses pengurutan data!

Insertion Sort

Perhatikan kode berikut :

d:\insertio.cpp

```
#include "stdio.h"

void main(){

   int  A[5]={3,4,1,2,8},i,j,tampung;

   printf("Sebelum sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }

   for (i=1;i<5;i++){
      tampung=A[i];
      j=i-1;

      while (A[j]>tampung && j>=0){
         A[j+1] = A[j];
         j--;
      }
      A[j+1]=tampung;
   }

   printf("\n\nSetelah sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }
}
```

Task 4

1. Apa yang dilakukan program diatas ?

2. Lakukan untuk pengurutan sebaliknya!
3. Apa fungsi **tampung**?
4. Buat program agar *user* bisa inputkan data secara dinamis, baik untuk *ascending*, maupun *descending*!
5. Tambahkan kode agar *user* dapat melihat proses pengurutan data!

Task 5

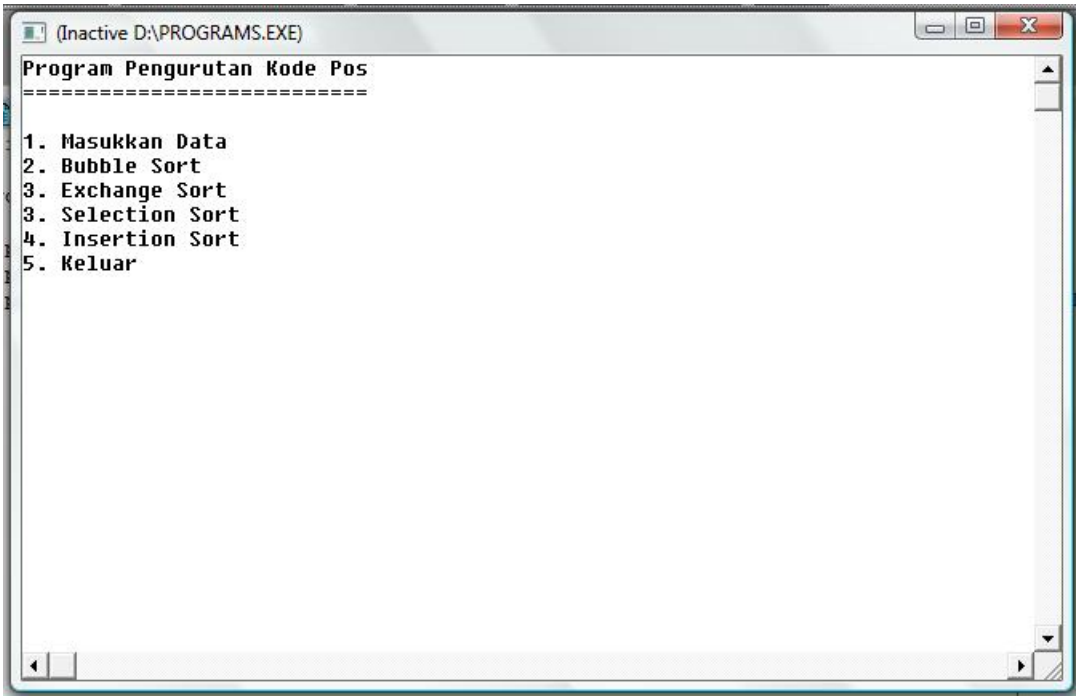
```
#include "stdio.h"
#include "iostream.h"

typedef struct surat{
int kodepos;
char pengirim[20];
char penerima[20];
char kota[20];
};

void main(){

   int i,j,tampung;
   surat suratku[5];

   for (i=0;i<5;i++){
      printf("\nSurat %i\n",i+1);
      printf("Kode pos : ");
      scanf("%i",&suratku[i].kodepos);
      printf("Pengirim : ");
      cin.getline(suratku[i].pengirim,20);
      printf("Penerima : ");
      cin.getline(suratku[i].penerima,20);
      printf("Kota : ");
      cin.getline(suratku[i].kota,20);
   }

   for (i=1;i<5;i++){
      for(j=5-1;j>=i;j--){
         if (suratku[j].kodepos<suratku[j-1].kodepos){
   }

   printf("\n\nSurat sesudah diurutkan\n");
   for (i=0;i<5;i++){
      printf("\n\nSurat %i\n",i+1);
      printf("\nKode pos : %i ",suratku[i].kodepos);
      printf("\nPengirim : %s ",suratku[i].pengirim);
      printf("\nPenerima : %s ",suratku[i].penerima);
      printf("\nKota : %s ",suratku[i].kota);
   }

}
```

1. Apa yang menjadi kesalahan pada program diatas?

2. Coba anda perbaiki!

Task 6

1. Gabungkan 4 sorting diatas menjadi 1 program menu pilihan untuk mengurutkan data kode pos.



2. User pertama kali harus memasukkan data terlebih dahulu, apabila tidak ada data, maka pengurutan data tidak bisa berjalan.
3. Data yang masuk hanya bisa berupa angka.
4. Semua metode mengurutkan secara ascending.
5. Pada tiap metode, user dapat melihat proses pengurutan data.

TAMBAHAN

Quick Sort

Quick sort merupakan algoritma kedua tercepat setelah merge sort. Quick sort disebut juga *divide and conquer algorithm.* Quick sort memiliki average case $_{n}\text{Log}_{n}$

Langkah-langkah quick sort :

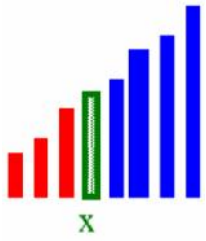
1. **SELECT**

   Pilih sebuah element , elemen ini kita sebut *pivot*

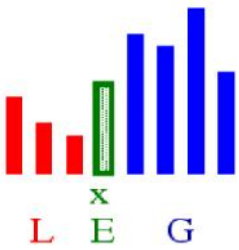
2. **DIVIDE**

   Kemudian pisahkan data menjadi 2 bagian, bagian yang **lebih kecil** dari pivot dan bagian yang **lebih besar** dari pivot

3. **RECUR and CONQUER**

   Kemudian 2 bagian tersebut diurutkan secara rekursif dengan memanggil fungsi itu sendiri, misal **quicksort()**

Perhatikan kode dibawah ini dan silakan di coba deprogram anda masing-masing:

```c
#include <stdio.h>

void quicksort (int *a, int lo, int hi)
{
int m = lo, n = hi, h;
int x = a[(lo + hi) / 2];

do {
while (a[m] < x) {
++m;
}
while (a[n] > x) {
--n;
}
if (m <= n) {
h = a[m];
a[m] = a[n];
a[n] = h;
++m;
--n;
}
} while (m <= n);

if (lo < n) {
quicksort(a, lo, n);
}
if (m < hi) {
quicksort(a, m, hi);
}
}

#define MAX 5

int main (void)
{
int data[MAX] = { 3, 4, 1, 2, 8 };
int x;

printf("sebelum disortir\n");

for (x = 0; x < MAX; ++x) {
printf("%d ", data[x]);
printf("\n");
quicksort(data, 0, MAX - 1);
printf("setelah disortir\n");
for (x = 0; x < MAX; ++x) {
printf("%d ", data[x]);
}
printf("");

return(0);
}
```

Task 7

1. Bagian mana pada program yang menentukan **pivot** ?
2. Bagian mana pada program yang memisahkan array menjadi 2 bagian ?

3. Bagian mana pada program yang melakukan fungsi rekursif ?

SOAL-SOAL

1. Soal Dan Pembahasan

- **Bubble Sort**

```
printf("\n");
for (i=10-1;i>=0;i--)
{
    for (j=0;j<i;j++)
    {
        for(k=0;k<10;k++)
        {
            printf("%i ",A[k]);
        }
        printf("\n");
        if(A[j]>A[j+1])
        {
            tampung=A[j];
            A[j]=A[j+1];
            A[j+1]=tampung;
        }
    }
    printf("\n");
}
```

- **Exchange Sort**

```
printf("\n");
for (i=0;i<9;i++)
{
    for (j=i+1;j<10;j++)
    {
        for(k=0;k<10;k++)
        {
            if(k==i)printf("|");
            printf("%i",A[k]);
            if(k==i)printf("|");
            printf(" ");
        }
        printf("\n");
        if(A[i]>A[j])
        {
            tampung=A[i];
            A[i]=A[j];
            A[j]=tampung;
        }
    }
    printf("\n");
}
```

- **Selection Sort**

```
printf("\n");
for (i=0;i<9;i++)
{
    pos=i;
    for (j=i+1;j<10;j++)
    {
        for(k=0;k<10;k++)
        {
            printf("%i ",A[k]);
        }
        printf("\tpos : %i\n",pos);
        if(A[pos]>A[j])
        {
            pos=j;
        }
    }
    printf("\n");
    if(i!=pos)
    {
        tampung=A[i];
        A[i]=A[pos];
        A[pos]=tampung;
    }

}
```

- **Insertion Sort**

```
for (i=0+1;i<10;i++)
{
    v=A[i];
    j=i-1;
    while (A[j]>v && j>=0)
    {
        A[j+1]=A[j];
        j--;
    }
    A[j+1]=v;
}
```

- Sebagai latihan keempat sorting di atas, lakukan:
  - Ubah code jadi sorting DESCENDING
  - Ubah code jadi inputan dinamis!(jumlah data tidak lagi 10)

2. Soal Tanpa Pembahasan

- **Search then Sort**

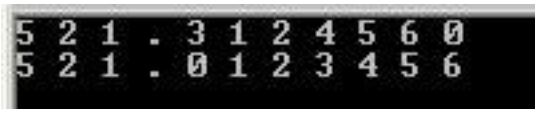
  Aio buat program yang seru-seru!
  Input nya sebuah deret angka yang mengandung tanda titik di dalam nya. Nah, kalo titik nya udah ketemu, terus semua angka di belakang nya di sorting jadi ascending. Nih contoh input output nya:
  Input :

  Banyak data : **10**

4 2 3 5 1 . 8 7 9 6

Output :

4 2 3 5 1 . 6 7 8 9



Note : tanda titik hanya akan ada 1 dalam 1 testcase.

- **2D Sort**

  Buatlah program untuk melakukan sorting array 2 dimensi!

  Input :

  m : **3**

  n : **2**

  3 2 1

  5 4 3

  Output:

  1 2 3

  3 4 5

3. Soal Take Home (***Take Home Test maybe?*)

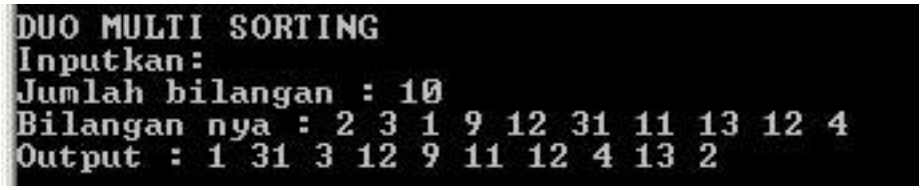
   Seorang yang amat sangat cerdas mendapatkan tugas yang menarik, dia di haruskan membuat sebuah program dengan menu:

   a. Duo Multi Sorting

   Menu pertama adlaah program "duo multi sorting" dengan contoh sorting:

   Barisan angka yang belum di sorting: **4 2 5 1 8 7 10 9 6**

   Barisan angka hasil multi sorting : **1 10 3 8 5 6 7 4 9 2**

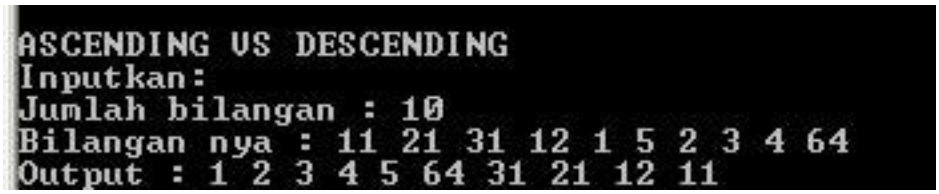


   Kondisi-kondisi yang menjadi batasan adalah:

   - Panjang data tidak akan pernah lebih dari 20
   - Algoritma sorting yang digunakan **BEBAS!**

   b. Ascending vs Descending

Menu kedua adalah program “Ascending vs Descending” dengan contoh sorting:
Barisan angka yang belum di sorting : **4 2 3 5 1 8 7 10 9 6 12 14 15 13 11**
Barisan angka yang sudah di sorting : **1 2 3 4 5 10 9 8 7 6 11 12 13 14 15**

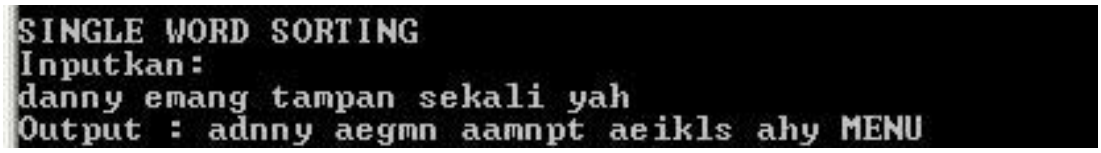


Kondisi-kondisi yang menjadi batasan adalah:

- Panjang data tidak akan pernah lebih dari 30
- Setiap 5 digit, terjadi perubahan sorting, **CEK POLA!**
- Algoritma sorting yang digunakan harus **EXCHANGE SORT!**

c. Single Word Sorting

Menu ketiga adalah program “Single Word Sorting” dengan contoh sorting :
Kalimat yang belum disorting : **danny emang tampan sekali**
Kalimat sesudah disorting : **adnny aegmn aamnpt aeikls**



Kondisi-kondisi yang menjadi batasan adalah:

- Panjang kalimat tidak pernah lebih dari 30 suku(sudah termasuk spasi dan kawan” nya)
- Algoritma sorting yang digunakan harus **SELECTION SORT!**

Nah, karena otak nya sudah tidak lagi cerdas, dia meminta bantuan pada kita yang masih lebih cerdas daripada dia…:D

Daftar Pustaka

- http://en.wikipedia.org/wiki/Sorting_algorithm
- *lecturer.eepis-its.edu/~entin/Struktur%20Data/Minggu%2011/Minggu%2011%20**Quick**.pdf*
- http://www.informatika.org/~rinaldi/Stmik/2005-2006/Makalah2006/MakalahStmik2006-27.pdf
- http://lecturer.ukdw.ac.id/anton/strukdat.php

Sorting Algorithm

Bubble Sort -- Selection Sort -- Insertion Sort -- Exchange Sort

Sorting Algorithm

Salah satu bagian penting dari struktur data adalah **sorting** atau pengurutan data. Ada banyak sekali Algoritma pengurutan data di dunia komputer, yatu : **bubble sort, selection sort, insertion sort, exchange sort,** quick sort, merge sort, dll.

Setiap algoritma memiliki kelebihan dan kekurangan masing – masing, kita akan mempelajari cukup 4 algoritma saja, yaitu bubble sort, selection sort, insertion sort, dan exchange sort. Bila ingin mempelajari algoritma yang lain silakan berkunjung ke www.wikipedia.org .

Bubble Sort

Perhatikan kode berikut :

d:\bubble.cpp

```
#include "stdio.h"

void main(){

   int  A[5]={3,4,1,2,8},i,j,tampung;

   printf("Sebelum sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }

   for (i=1;i<5;i++){
      for(j=5-1;j>=i;j--){
         if (A[j]<A[j-1]){
            tampung=A[j];
            A[j]=A[j-1];
            A[j-1]=tampung;
         }
      }
   }

   printf("\n\nSetelah sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }
}
```

Task 1

6. Apa yang dilakukan program diatas ?
7. Lakukan untuk pengurutan sebaliknya!

8. Pengurutan diatas dilakukan dari depan atau belakang? Buat program untuk sebaliknya!
9. Buat program agar *user* bisa inputkan data secara dinamis, baik untuk *ascending*, maupun *descending*!
10. Tambahkan kode agar *user* dapat melihat proses pengurutan data!

    Note : Ascending adalah pengurutan data dari terkecil menuju terbesar, sedangkan descending adalah pengurutan dari data terbesar menuju terkecil.

Exchange Sort

Perhatikan kode berikut :

d:\exchange.cpp

```
#include "stdio.h"

void main(){

   int  A[5]={3,4,1,2,8},i,j,tampung;

   printf("Sebelum sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }

   for (i=0;i<5-1;i++){
      for(j=i+1;j<5;j++){
         if (A[i]<A[j]){
            tampung=A[i];
            A[i]=A[j];
            A[j]=tampung;
         }
      }
   }

   printf("\n\nSetelah sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }
}
```

Task 2

6. Apa yang dilakukan program diatas ?

7. Lakukan untuk pengurutan sebaliknya!
8. Buat program agar *user* bisa inputkan data secara dinamis, baik untuk *ascending*, maupun *descending*!
9. Tambahkan kode agar *user* dapat melihat proses pengurutan data!
10. Apa perbedaan antara exchange sort dan bubble sort!

Selection Sort

Perhatikan kode berikut :

d:\selectio.cpp

```
int  A[5]={3,4,1,2,8},i,j,tampung,pos;

printf("Sebelum sorting : \n");
for (i=0;i<5;i++){
   printf("%i ",A[i]);
}

for (i=0;i<5-1;i++){
   pos=i;
   for(j=i+1;j<5;j++){
      if (A[j]<A[pos]){
         pos=j;
      }
   }
   if (pos != i){
      tampung=A[pos];
      A[pos]=A[i];
      A[i]=tampung;
   }
}

printf("\n\nSetelah sorting : \n");
for (i=0;i<5;i++){
   printf("%i ",A[i]);
}
}
```

Task 3

6. Apa yang dilakukan program diatas ?
7. Lakukan untuk pengurutan sebaliknya!
8. Apa fungsi **pos**?
9. Buat program agar *user* bisa inputkan data secara dinamis, baik untuk *ascending*, maupun *descending*!
10. Tambahkan kode agar *user* dapat melihat proses pengurutan data!

Insertion Sort

Perhatikan kode berikut :

d:\insertio.cpp

```
#include "stdio.h"

void main(){

   int  A[5]={3,4,1,2,8},i,j,tampung;

   printf("Sebelum sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }

   for (i=1;i<5;i++){
      tampung=A[i];
      j=i-1;

      while (A[j]>tampung && j>=0){
         A[j+1] = A[j];
         j--;
      }
      A[j+1]=tampung;
   }

   printf("\n\nSetelah sorting : \n");
   for (i=0;i<5;i++){
      printf("%i ",A[i]);
   }
}
```

Task 4

6. Apa yang dilakukan program diatas ?
7. Lakukan untuk pengurutan sebaliknya!
8. Apa fungsi **tampung**?
9. Buat program agar *user* bisa inputkan data secara dinamis, baik untuk *ascending*, maupun *descending*!
10. Tambahkan kode agar *user* dapat melihat proses pengurutan data!

Task 5

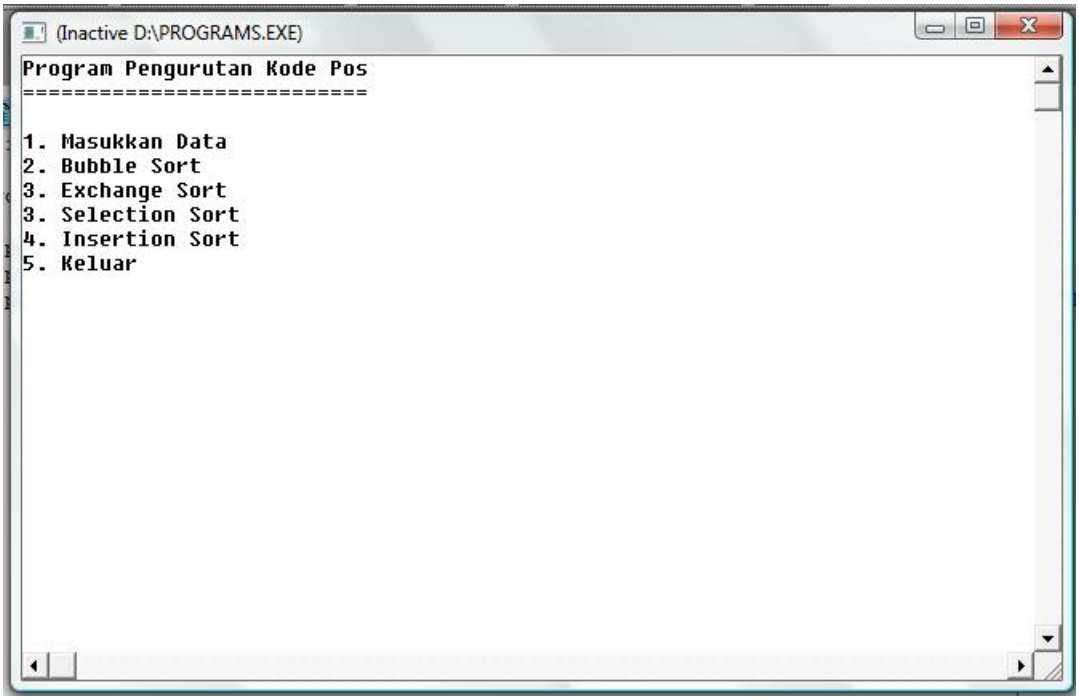
```
#include "stdio.h"
#include "iostream.h"

typedef struct surat{
int kodepos;
char pengirim[20];
char penerima[20];
char kota[20];
};

void main(){

   int i,j,tampung;
   surat suratku[5];

   for (i=0;i<5;i++){
      printf("\nSurat %i\n",i+1);
      printf("Kode pos : ");
      scanf("%i",&suratku[i].kodepos);
      printf("Pengirim : ");
      cin.getline(suratku[i].pengirim,20);
      printf("Penerima : ");
      cin.getline(suratku[i].penerima,20);
      printf("Kota : ");
      cin.getline(suratku[i].kota,20);
   }

   for (i=1;i<5;i++){
      for(j=5-1;j>=i;j--){
         if (suratku[j].kodepos<suratku[j-1].kodepos){
   }

   printf("\n\nSurat sesudah diurutkan\n");
   for (i=0;i<5;i++){
      printf("\n\nSurat %i\n",i+1);
      printf("\nKode pos : %i ",suratku[i].kodepos);
      printf("\nPengirim : %s ",suratku[i].pengirim);
      printf("\nPenerima : %s ",suratku[i].penerima);
      printf("\nKota : %s ",suratku[i].kota);
   }

}
```

3. Apa yang menjadi kesalahan pada program diatas?

4. Coba anda perbaiki!

Task 6

6. Gabungkan 4 sorting diatas menjadi 1 program menu pilihan untuk mengurutkan data kode pos.



7. User pertama kali harus memasukkan data terlebih dahulu, apabila tidak ada data, maka pengurutan data tidak bisa berjalan.
8. Data yang masuk hanya bisa berupa angka.
9. Semua metode mengurutkan secara ascending.
10. Pada tiap metode, user dapat melihat proses pengurutan data.

TAMBAHAN

Quick Sort

Quick sort merupakan algoritma kedua tercepat setelah merge sort. Quick sort disebut juga *divide and conquer algorithm.* Quick sort memiliki average case $_{n}Log_{n}$

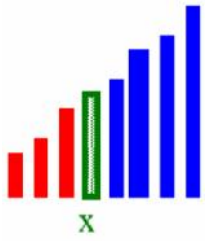
Langkah-langkah quick sort :

4. **SELECT**

   Pilih sebuah element , elemen ini kita sebut *pivot*

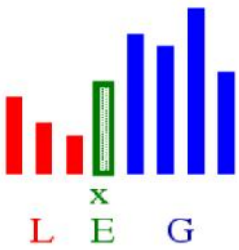
5. **DIVIDE**

   Kemudian pisahkan data menjadi 2 bagian, bagian yang **lebih kecil** dari pivot dan bagian yang **lebih besar** dari pivot



6. **RECUR and CONQUER**

   Kemudian 2 bagian tersebut diurutkan secara rekursif dengan memanggil fungsi itu sendiri, misal **quicksort()**



Perhatikan kode dibawah ini dan silakan di coba deprogram anda masing-masing:

```c
#include <stdio.h>

void quicksort (int *a, int lo, int hi)
{
int m = lo, n = hi, h;
int x = a[(lo + hi) / 2];

do {
while (a[m] < x) {
++m;
}
while (a[n] > x) {
--n;
}
if (m <= n) {
h = a[m];
a[m] = a[n];
a[n] = h;
++m;
--n;
}
} while (m <= n);

if (lo < n) {
quicksort(a, lo, n);
}
if (m < hi) {
quicksort(a, m, hi);
}
}

#define MAX 5

int main (void)
{
int data[MAX] = { 3, 4, 1, 2, 8 };
int x;

printf("sebelum disortir\n");

for (x = 0; x < MAX; ++x) {
printf("%d ", data[x]);
printf("\n");
quicksort(data, 0, MAX - 1);
printf("setelah disortir\n");
for (x = 0; x < MAX; ++x) {
printf("%d ", data[x]);
}
printf("");

return(0);
}
```

Task 7

4. Bagian mana pada program yang menentukan **pivot** ?

5. Bagian mana pada program yang memisahkan array menjadi 2 bagian ?

6. Bagian mana pada program yang melakukan fungsi rekursif ?

SOAL-SOAL

4. Soal Dan Pembahasan

- **Bubble Sort**

```
printf("\n");
for (i=10-1;i>=0;i--)
{
    for (j=0;j<i;j++)
    {
        for(k=0;k<10;k++)
        {
            printf("%i ",A[k]);
        }
        printf("\n");
        if(A[j]>A[j+1])
        {
            tampung=A[j];
            A[j]=A[j+1];
            A[j+1]=tampung;
        }
    }
    printf("\n");
}
```

- **Exchange Sort**

```
printf("\n");
for (i=0;i<9;i++)
{
    for (j=i+1;j<10;j++)
    {
        for(k=0;k<10;k++)
        {
            if(k==i)printf("|");
            printf("%i",A[k]);
            if(k==i)printf("|");
            printf(" ");
        }
        printf("\n");
        if(A[i]>A[j])
        {
            tampung=A[i];
            A[i]=A[j];
            A[j]=tampung;
        }
    }
    printf("\n");
}
```

- **Selection Sort**

```
printf("\n");
for (i=0;i<9;i++)
{
    pos=i;
    for (j=i+1;j<10;j++)
    {
        for(k=0;k<10;k++)
        {
            printf("%i ",A[k]);
        }
        printf("\tpos : %i\n",pos);
        if(A[pos]>A[j])
        {
            pos=j;
        }
    }
    printf("\n");
    if(i!=pos)
    {
        tampung=A[i];
        A[i]=A[pos];
        A[pos]=tampung;
    }

}
```

- **Insertion Sort**

```
for (i=0+1;i<10;i++)
{
    v=A[i];
    j=i-1;
    while (A[j]>v && j>=0)
    {
        A[j+1]=A[j];
        j--;
    }
    A[j+1]=v;
}
```

- Sebagai latihan keempat sorting di atas, lakukan:
  - Ubah code jadi sorting DESCENDING
  - Ubah code jadi inputan dinamis!(jumlah data tidak lagi 10)

5. Soal Tanpa Pembahasan

- **Search then Sort**

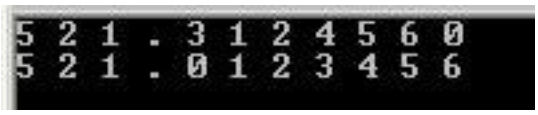
  Aio buat program yang seru-seru!
  Input nya sebuah deret angka yang mengandung tanda titik di dalam nya. Nah, kalo titik nya udah ketemu, terus semua angka di belakang nya di sorting jadi ascending. Nih contoh input output nya:
  Input :

  Banyak data : **10**

4 2 3 5 1 . 8 7 9 6

Output :

4 2 3 5 1 . 6 7 8 9



Note : tanda titik hanya akan ada 1 dalam 1 testcase.

- **2D Sort**

  Buatlah program untuk melakukan sorting array 2 dimensi!

  Input :

  m : **3**

  n : **2**

  3 2 1

  5 4 3

  Output:

  1 2 3

  3 4 5

6. Soal Take Home (***Take Home Test maybe?*)

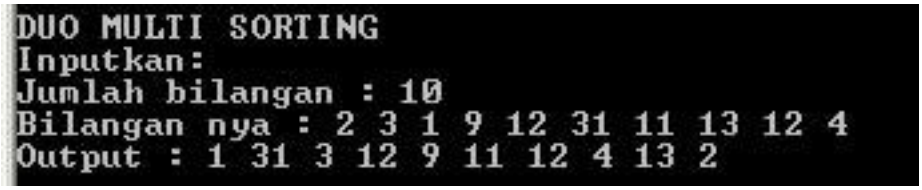
   Seorang yang amat sangat cerdas mendapatkan tugas yang menarik, dia di haruskan membuat sebuah program dengan menu:

   d. Duo Multi Sorting

   Menu pertama adlaah program "duo multi sorting" dengan contoh sorting:

   Barisan angka yang belum di sorting: **4 2 5 1 8 7 10 9 6**

   Barisan angka hasil multi sorting : **1 10 3 8 5 6 7 4 9 2**

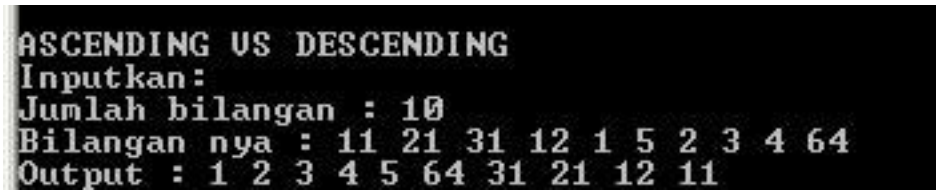


   Kondisi-kondisi yang menjadi batasan adalah:

   - Panjang data tidak akan pernah lebih dari 20
   - Algoritma sorting yang digunakan **BEBAS!**

   e. Ascending vs Descending

Menu kedua adalah program “Ascending vs Descending” dengan contoh sorting:
Barisan angka yang belum di sorting : **4 2 3 5 1 8 7 10 9 6 12 14 15 13 11**
Barisan angka yang sudah di sorting : **1 2 3 4 5 10 9 8 7 6 11 12 13 14 15**

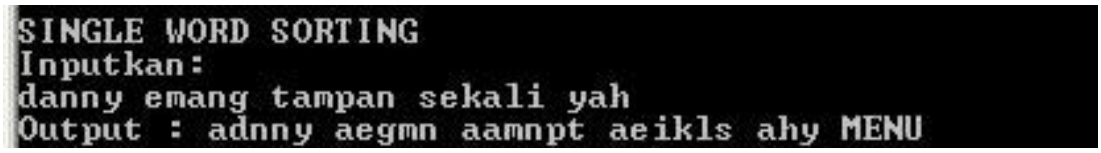


Kondisi-kondisi yang menjadi batasan adalah:

- Panjang data tidak akan pernah lebih dari 30
- Setiap 5 digit, terjadi perubahan sorting, **CEK POLA!**
- Algoritma sorting yang digunakan harus **EXCHANGE SORT!**

f. Single Word Sorting

Menu ketiga adalah program “Single Word Sorting” dengan contoh sorting :
Kalimat yang belum disorting : **danny emang tampan sekali**
Kalimat sesudah disorting : **adnny aegmn aamnpt aeikls**



Kondisi-kondisi yang menjadi batasan adalah:

- Panjang kalimat tidak pernah lebih dari 30 suku(sudah termasuk spasi dan kawan” nya)
- Algoritma sorting yang digunakan harus **SELECTION SORT!**

Nah, karena otak nya sudah tidak lagi cerdas, dia meminta bantuan pada kita yang masih lebih cerdas daripada dia…:D

Daftar Pustaka

- http://en.wikipedia.org/wiki/Sorting_algorithm
- *lecturer.eepis-its.edu/~entin/Struktur%20Data/Minggu%2011/Minggu%2011%20**Quick**.pdf*
- http://www.informatika.org/~rinaldi/Stmik/2005-2006/Makalah2006/MakalahStmik2006-27.pdf
- http://lecturer.ukdw.ac.id/anton/strukdat.php

STACK dan QUEUE

PENGANTAR

Definisi

Stack disebut juga tumpukan dimana data hanya dapat dimasukkan dan diambil dari satu sisi.
Karena itu, stack bersifat LIFO(Last In First Out).

Operasi yang dapat dilakukan stack adalah:

1. Menambah (push)
2. Mengambil (pop)
3. megecek apakah stack penuh (isFull)
4. mengecek apakah stack kosong (isEmpty)
5. membersihkan stack (clear).
6. Mencetak isi stack (print)

Operasi-operasi stack

Saat ini, kita akan mencoba membuat stack dan operasi-operasi yang dapat dilakukannya.

1. Mendefinisikan stack dengan menggunakan struct

```
typedef struct stack //       Mendefinisikan stack dengan menggunakan struct
{
int top;
char data [15][20];     // menampung 15 data dengan jumlah string max 20 huruf
};
```

2. Mendefinisikan max_stack untuk maksimum isi stack
   `#define max_stack 15`

3. Membuat variable array sebagai implementasi stack
   `stack tumpuk;`

4. Mendeklarasikan operasi-operasi/fungsi yang dapat dilakukan stack.
   a. Push (menginputkan data pada stack)

```
void push(char d[20])
{
        tumpuk.top++;
        strcpy(tumpuk.data[tumpuk.top],d);
        printf("data berhasil dimasukkan");
}
```

b. Pop (mengambil data pada stack)

```
void pop()
{
        printf ("data %s terambil",tumpuk.data[tumpuk.top]);
        tumpuk.top--;
}
```

c. IsFull(megecek apakah stack penuh)

```
int isFull()
{
        if (tumpuk.top==max_stack-1)
                return 1;
        else
                return 0;
}
```

d. isEmpty(mengecek apakah stack kosong)

```
int isEmpty()
{
        if (tumpuk.top==-1)
                return 1;
        else
                return 0;
}
```

e. clear (membersihkan seluruh isi stack)

```
void clear()
{
        tumpuk.top=-1;
        printf("semua data terhapus.");

}
```

f. print (mencetak seluruh isi stack)

```
void print()
{
        for (int i=tumpuk.top;i>=0;i--)
                printf ("%s\n",tumpuk.data[i]);
}
```

QUEUE

Queue disebut juga antrian dimana data masuk di satu sisi dan keluar di sisi yang lain. Karena itu, queue bersifat FIFO(First In First Out).

Saat ini, kita akan mencoba membuat queue dan operasi-operasi yang dapat dilakukannya.

Hal-hal yang perlu dilakukan untuk membuat queue yaitu

1. Mendefinisikan queue dengan menggunakan struct dimana kita perlu menggunakan variable head dan tail sebagai penanda pada stack.

```
typedef struct queue //       Mendefinisikan queue dengan menggunakan struct
{
int head;
int tail;
char data [15][20];     // menampung 15 data dengan jumlah string max 20 huruf
};
```

2. Mendefinisikan max untuk maksimum isi queue
   #define max 15

3. Membuat variable array sebagai implementasi queue
   queue antri;

4. Mendeklarasikan operasi-operasi/fungsi yang dapat dilakukan queue.
   a. Enqueue (menginputkan data pada queue)

```
void enqueue(char d[20])
{
        antri.head=0;
        antri.tail++;
        strcpy(antri.data[antri.tail],d);
        printf("data berhasil dimasukkan");
}
```

   b. Dequeue (mengambil data dari queue)

```
void dequeue()
{
        printf ("data %s terambil",antri.data[antri.head]);
        for (int i=antri.head;i<=antri.tail;i++)
                strcpy (antri.data[i],antri.data[i+1]);
        antri.tail--;
}
```

   c. isEmpty (mengecek apakah antrian kosong)

```
int isEmpty()
{
        if (antri.tail==-1)
        {
                antri.head=-1;
                return 1;
        }
        else
                return 0;
}
```

d. isFull (mengecek apakah antrian penuh)

```
int isFull()
{
        if (antri.tail==max-1)
                return 1;
        else
                return 0;
}
```

e. clear (membersihkan seluruh isi antrian)

```
void clear()
{
        antri.head=antri.tail=-1;
        printf("semua data terhapus.");

}
```

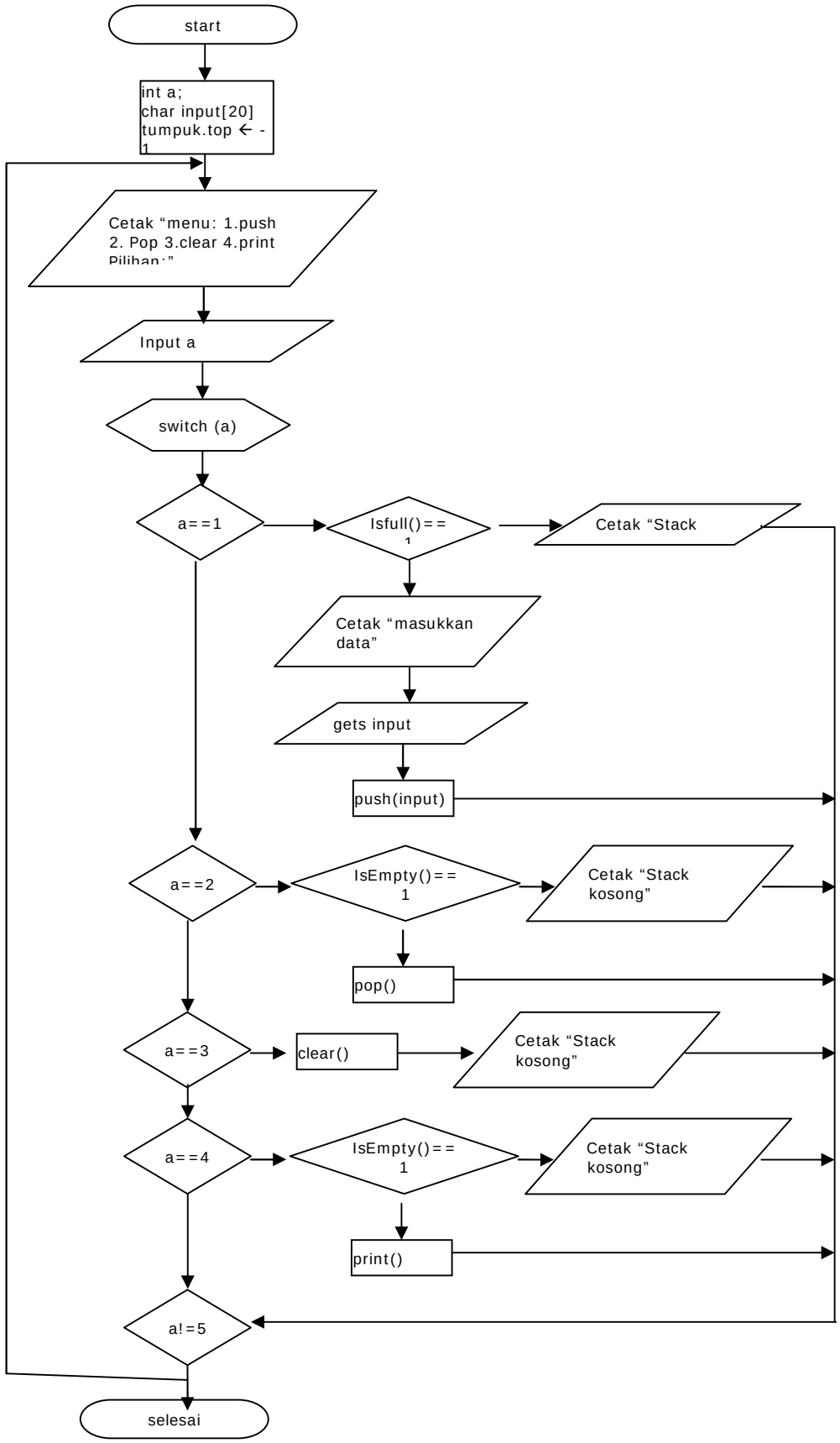
f. Print(mencetak seluruh isi antrian)

```
void print()
{
        for (int i=0;i<=antri.tail;i++)
                printf ("%s\n",antri.data[i]);
}
```

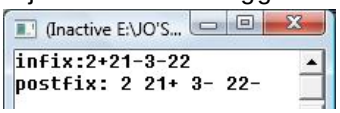
SOAL LATIHAN:

Guided STACK:

1. Dari flowchart dibawah ini, buatlah fungsi utama untuk menjalankan stack diatas dengan menggunakan menu.

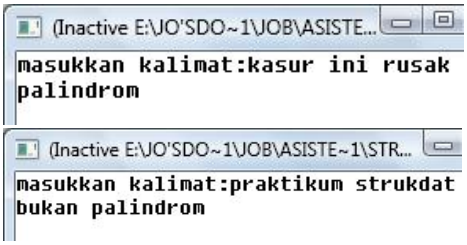


2. Buatlah program pengubah infix ke postfix dengan asumsi operator hanya tambah(+) dan kurang(-) saja. buatlah menggunakan stack!

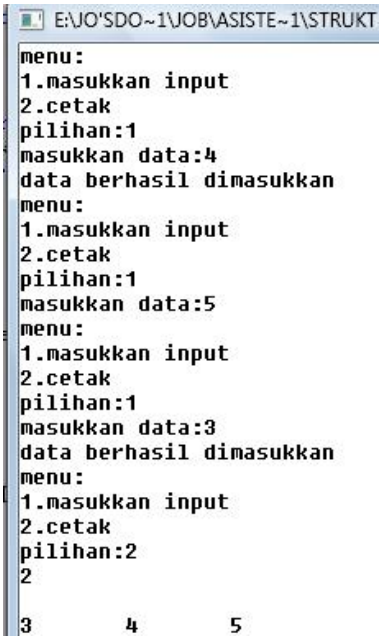


Unguided STACK:

1. Buatlah program pengecekan bilangan palindrom menggunakan stack!



2. Buatlah program pengurutan menggunakan stack! Sehingga input yang kita masukkan selalu urut.
   Hint: buatlah menggunakan 2buah stack!



TAKEHOME STACK!

1. Kembangkan soal unguided nomor 2 untuk mengurutkan kata!
2. Kembangkan soal guided nomor 2 dengan operator bertingkat! Gunakan 2 buah stack. Contoh tersedia di bawah ini:
   Infix: a+b*c-d

Stack (kosong) dan Postfik (kosong)
Scan a
Postfik: a
Scan +
Stack: +
Scan b
Postfik: ab
Scan *, karena ToS (+) < *, maka add ke Stack
Stack: +*
Scan c
Postfik: abc
Scan –, karena * > -, maka pop Stack, dan add ke Postfik
Stack: +
Postfik: abc*
Karena + <= -, maka pop Stack, dan add ke Postfik, karena Stack kosong, maka push – ke stack
Stack: -
Postfik: abc*+
Scan d
Postfik: abc*+d
Karena sudah habis, push ToS stack ke Posfix

Modul I

Modul I

Kunci jawaban soal GUIDED STACK:

No.1:

Fungsi utamanya:

```
void main()
{
	int a;
	char input[20];
	tumpuk.top=-1;
do {
	printf ("menu:\n1.push\n2.Pop\n3.clear\n4.print\n5.terminate\npilihan:");
	scanf ("%i",&a);
	fflush(stdin);
	switch (a)
	{
	case 1:
		if (isFull()==1)
			printf("stack penuh.\n");
		else
			{
			printf("masukkan data:");
			gets(input);
			fflush(stdin);
			push(input);
			}
		break;
	case 2:
		if (isEmpty()==1)
			{
				printf("stack kosong");
			}
		else
			{
				pop();
			}
		break;
	case 3:
			clear();
			printf("stack kosong\n");
			break;
	case 4:
		if (isEmpty()==1)
			printf("stack kosong");
		else
			print();
	}
}
 while (a!=5);
}
```

No.2:

Jangan lupa sesuaikan fungsi pop, dan push yang sudah ada dengan kontex penggunaannya dan juga tipe datanya!

```
void main()
{
	int a;
	char input[20];
	tumpuk.top=-1;
	printf("infix:");
	gets(input);
	a=strlen(input);
	printf("%c",input[0]);
	for (int i=1;i<a;i++)
	{
		if (input[i]=='+' || input[i]=='-' )
		{
		push(input[i]);
		}
		else
		{
			printf("%c",input[i]);
			if(input[i+1]=='+' || input[i+1]=='-'|| i+1==a)
			{
				pop();
			}
		}

	}
}
```

Modul I

Latihan Terbimbing Queue

1. Buatlah fungsi utama untuk menjalankan queue pada modul dengan menggunakan menu.(untuk alurnya, hampir sama dengan flowchart untuk stack diatas. Hanya saja fungsi diganti dengan fungsi-fungsi queue.)
2. Carilah total, rata-rata, nilai terkecil, nilai terbesar dari nilai yang telah di inputkan pada queue

Latihan Mandiri Kelas Queue

1. Tambahkanlah function untuk mencari suatu elemen dalam queue & stack
2. Tambahkan function untuk mengedit suatu elemen dalam queue & stack
3. Buatlah suatu fungsi untuk mencetak elemen stack ganjil
4. Buatlah suatu fungsi untuk mencetak elemen queue genap

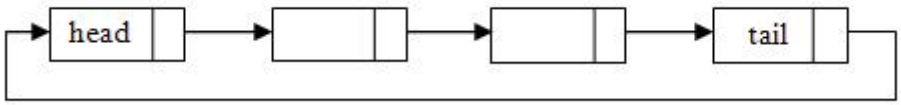
Latihan Mandiri Rumah /Tes Queue

1. Buatlah program untuk membuat kalkulator scientific!
2. Buatlah program untuk membalikkan kata yang berguna untuk membalik susunan elemen dalam queue Q dengan menggunakan bantuan sebuah stack .
3. Buatlah program untuk menentukan apakah kata yang dimasukan merupakan palindrome atau bukan.

Single Linked List Circular

(Senarai Berantai Tunggal Berputar)

Pemahaman akan Single Linked List Circular tidak jauh berbeda dengan Single Linked List non Circular, hanya saja pada Single Linked List Circular, gerbong terakhir (ekor) akan terhubung dengan gerbong depan (kepala).



Proses pembuatan Single Linked List Circular (SLLC)

Deklarasi Struct gerbong

```
typedef struct gerbong{
        int data;
        gerbong *next;
};
```

Modul I

Buat juga variable pointer bertipe gerbong

```
gerbong *head;
gerbong *tail;
gerbong *baru;
gerbong *hapus;
gerbong *bantu;
```

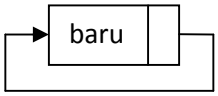
Pada main, inisialisasi head sebagai gerbong baru

```
head=new gerbong;
head=NULL;
```

Kita akan menggunakan isEmpty untuk mengecek

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

Bagaimana jika penambahan data depan dan data belakang?



1. Tambah depan

Modul I

```
void tambahdepan( int databaru ){
        baru=new gerbong;
        baru->data=databaru;
        baru->next=baru;

        if( isEmpty() ){
            head=tail=baru;
            head->next=head;
            tail->next=tail;
        }
        else{
            baru->next=head;
            head=baru;
            tail->next=head;
        }
        printf("data masuk\n");
}
```

Apakah ada bedanya dengan SLLNC?

2. Tambah belakang

```
void tambahbelakang( int databaru ){
        baru=new gerbong;
        baru->data=databaru;
        baru->next=baru;

        if( isEmpty() ){
            head=tail=baru;
            head->next=head;
            tail->next=tail;
        }
        else{
            tail->next=baru;
            tail=baru;
            tail->next=head;
        }
        printf("data masuk\n");
}
```

Apakah ada bedanya dengan SLLNC?

Latihan 1

Ubah fungsi tambah depan untuk data berikut: Nim, nama mahasiswa, dan ipk. Dengan Nim menambah secara otomatis. Sedangkan nama mahasiswa dan ipk diinputkan oleh user.

Bagaimana jika menampilan semua data?

Modul I

```
void cetak(){
    if( !isEmpty() ){
        bantu=head;
        printf(" %i ", bantu->data);
        bantu=bantu->next;
        while(bantu!=head){
            printf(" %i ", bantu->data);
            bantu=bantu->next;
        }
        printf("\n");
    }
    else{
        printf(“Data kosong!”);
    }
}
```

Untuk menampilkan semua data, kita hanya mengecek semua data sampai kembali ke data posisi awal (head). Jika telah kembali ke posisi head maka proses cetak dihentikan.

Bagaimana menghapus data depan dan data belakang?

1. Hapus depan

```
void hapusdepan(){
    int tampung;
    if( isEmpty()){
        printf("data belum ada");
    }
    else{
        tampung=head->data;
        if(head->next!=NULL){
            hapus=head;
            head=head->next;
            tail->next=head;
            delete hapus;
        }
        else{
            head=tail=NULL;
        }
    printf("Data %i Terhapus\n", tampung);
    }
}
```

Hapus depan mengharuskan kita untuk menyambung ulang tail ke head, dengan cara ini linked list akan selalu berputar.

2. Hapus belakang

```
void hapusbelakang(){
        int tampung;
        if( isEmpty() ){
                printf("data belum ada");
        }
        else{
                tampung=tail->data;
                if(head==tail){
                        head=tail=NULL;
                }
                else{
                        bantu=head;
                        hapus=tail;
                        while(bantu->next!=tail){
                                bantu=bantu->next;
                        }
                tail=bantu;
                tail->next=head;
                delete hapus;
        }
        printf("Data %i Terhapus\n", tampung);
        }
}
```

Sama halnya dengan hapus depan, hapus belakang juga mengharuskan data sebelum tail(bantu) untuk menyambung kembali dengan head.

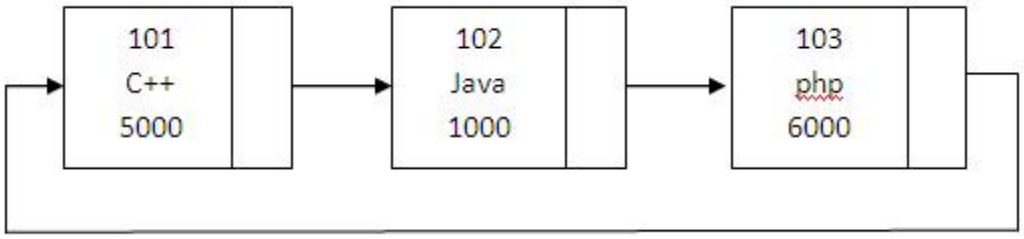
Latihan 2

Buat fungsi Hapus data. Data yang akan dihapus harus di inputkan oleh user.

EXERCISE

1. Buatlah fungsi tambah data yang dapat secara otomatis mengurutkan data yang diinputkan user. Data akan terurut secara ascending.
2. Buat sebuah Queue dengan menggunakan SLLC. Struck dari Queue ini berisi NoAntrian(int) dan Nama(String). Buat berupa menu yang berisi:
    1. Masuk data
    2. Liat data
    3. Keluar data
    4. Exit dari menu.
3. Buatlah SLLC untuk data buku seperti contoh dibawah ini:

Modul I



Buatlah menu untuk:

a. Menambah data buku (tambah belakang)
b. Cetak
c. Edit Harga(cari buku yang akan di edit berdasarkan nomer buku, lalu harganya saja yang diedit)
d. Exit

4. Buat converter dari array ke SLLC.
   a. Buat dalam bentuk menu (tambah data, lihat, dan exit)
   b. Tambah data merupakan sebuah string dan langsung di convert ke Link List
   c. Setiap Link List menampung sebuat karakter
   d. Setiap tambah data, tidak menghilangkan string yg diinputkan pertama. Contoh:

Inputan pertama:

```
E:\STUDY\ASDOS\STRUKD~1\NC
Menu :
1. Add
2. View
3. Dexit
==============
Pilihan : 1
maskkan string = bebek
```

Input Kedua:

```
E:\STUDY\ASDOS\STRUKD~1\
Menu :
1. Add
2. View
3. Dexit
==============
Pilihan : 1
maskkan string =  goreng
```

hasil:

```
E:\STUDY\ASDOS\STRUKD~1\NONAME00.EXE
Menu :
1. Add
2. View
3. Dexit
==============
Pilihan : 2
 b  e  b  e  k     g  o  r  e  n  g
```

Modul I

DOUBLE LINKED LIST

DOUBLE LINKED LIST

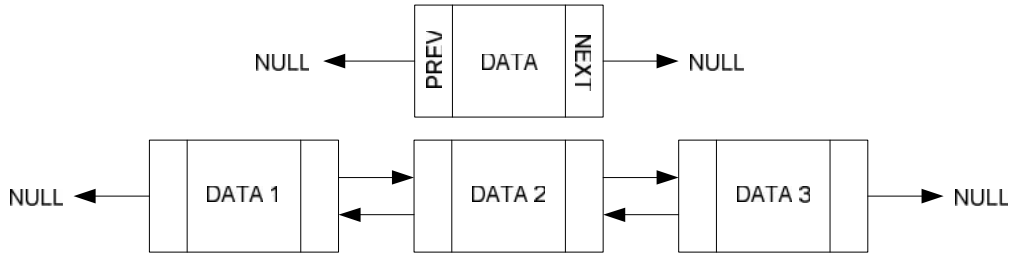
- Pada dasarnya, penggunaan Double Linked List hampir sama dengan penggunaan Single Linked List yang telah kita pelajari pada materi sebelumnya. Hanya saja Double Linked List menerapkan sebuah pointer baru, yaitu **prev**, yang digunakan untuk menggeser mundur selain tetap mempertahankan pointer **next**.
- 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



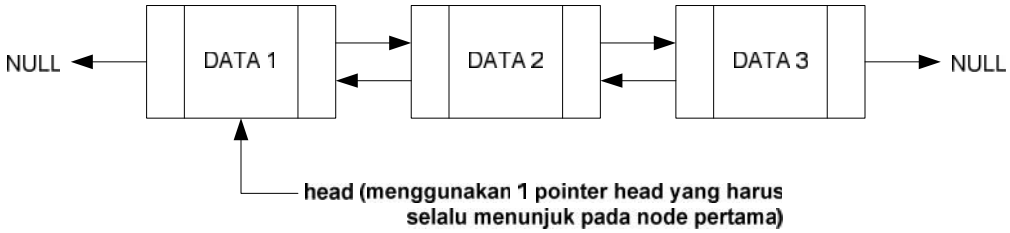
c. PEMBUATAN DLLNC

- **Deklarasi Node**

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

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

Modul I



- 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

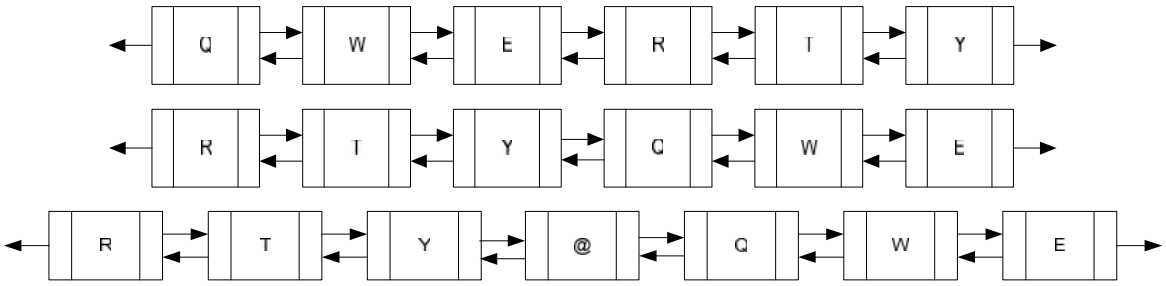
```
void cetak()                                         // menampilkan semua data
{
      TNode *bantu;
      bantu = head;                                 // letakan bantu pada head

      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 :**

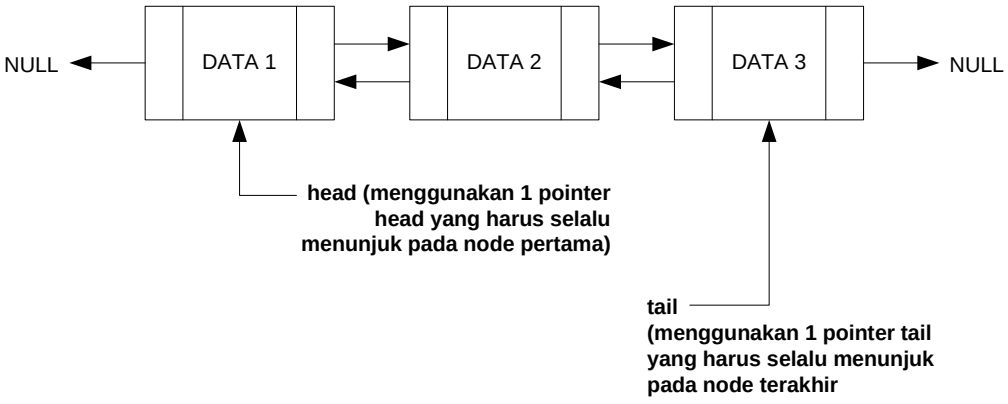
Modul I

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;
}
```

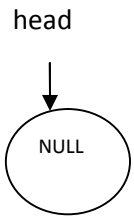
Contoh program :

- 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");
}
```

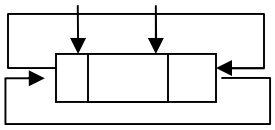
Penggambaran :



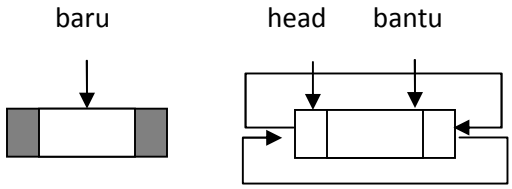
Setelah dibuat node baru dan jika diketahui head==NULL :

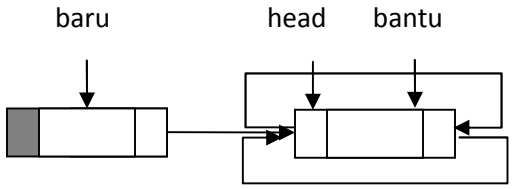
head baru

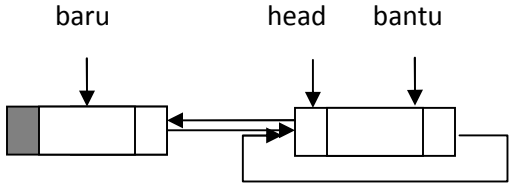
Modul I

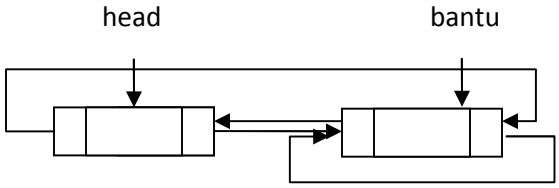


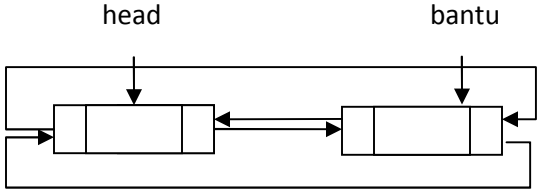
Bila kita membuat node baru lagi maka :











- 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 :

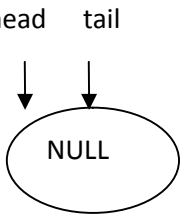
- Penambahan di depan

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

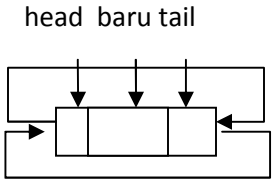
```
   if (isEmpty()==1) {
         head=baru;
         tail=head;
         head->next=head;
         head->prev=head;
         tail->next=tail;
         tail->prev=tail;
   }
   else {
         baru->next=head;
         head->prev=baru;
         head=baru;
         head->prev=tail;
         tail->next=head;
   }
```

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

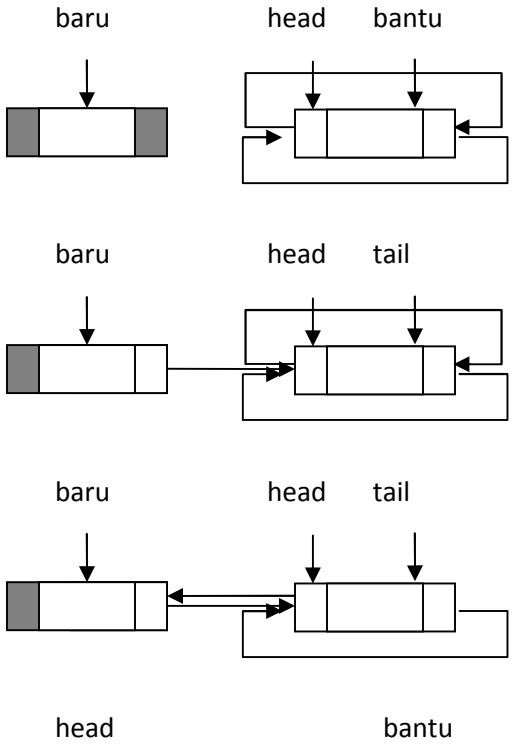
Penggambaran :



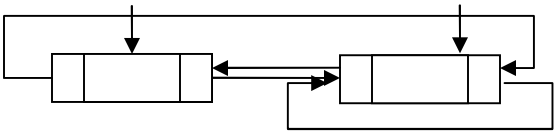
Setelah dibuat node baru dan jika diketahui tail==NULL :



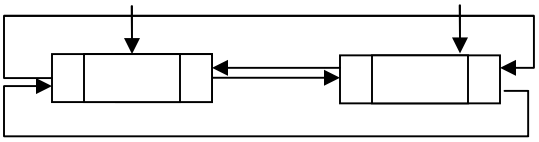
Bila kita membuat node baru lagi maka :



Modul I



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");
}
```

Modul I

- 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.

Modul I

- 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;
}
```

Modul I

```
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;
```

Modul I

```
 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

Modul I

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…

Fungsi Rekursif

Fungsi rekursif adalah suatu fungsi yang memanggil dirinya sendiri. Pada beberapa persoalan, fungsi rekursif sangat berguna karena mempermudah solusi. Namun demikian, fungsi rekursif juga memiliki kelemahan, yakni memungkinkan terjadinya overflow pada stack, yang berarti stack tidak lagi mampu menangani permintaan pemanggilan fungsi karena kehabisan memori(stack adalah area memori yang dipakai untuk variable lokal untuk mengalokasikan memori ketika suatu fungsi dipanggil. Oleh karena itu, jika bisa diselesaikan dengan metode iteratif, gunakanlah metode iteratif.

Bentuk umum fungsi rekursif.

```
return_data_type function_name(parameter_list)
```

Modul I

```
{
   ...
   function_name(...);
   ...
}
```

Contoh program

Contoh #1: Faktorial

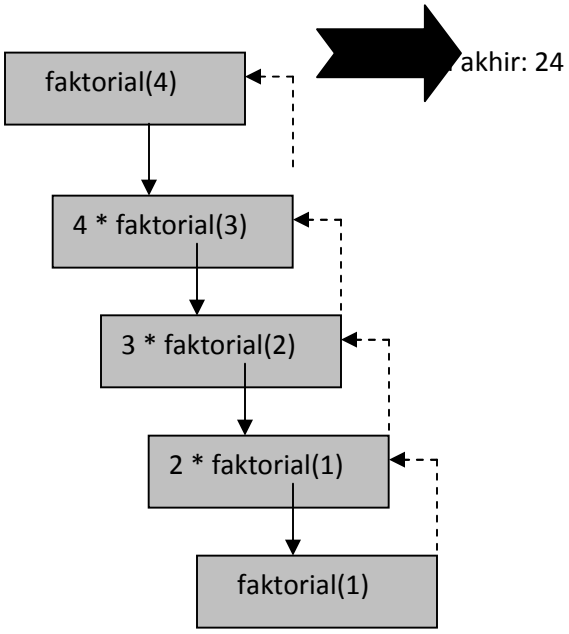
Fungsi rekursif dapat digunakan untuk menghitung faktorial. Berikut penjelasan beserta dengan contoh listing programnya.

Fungsi faktorial dapat dinyatakan dalam bentuk rekursif seperti berikut:

fak(n) = 1, untuk n = 0 atau n = 1

fak(n) = n x (n-1)!, untuk n > 0

Berikut contoh proses untuk perolehan hasil 4!.



Modul I

Berikut implementasi program dengan metode iteratif.

```
#include <stdio.h>


int fak(int n)
  {
int i,fak;


if(n == 0 || n == 1)
    return 1;
  else
  {
    temp = 1;
    for (i=1; i<=n; i++)
    fak = fak * i;
    return (fak);
  }
  void main()
  {
  int fakt;
  printf("Masukkan berapa faktorial : ");
  scanf("%d",&fakt);
  printf("Hasil faktorial dari adalah : %d\n", fak(fakt));
  }
```

Berikut implementasi program untuk proses faktorial dengan metode rekursif:

Modul I

```
#include <stdio.h>

int fak( int n )
{
    if (n == 0 || n == 1)
        return 1;
    else
        return n * fak(n-1);
}

void main()
{
      int n, hasil;
      printf("Masukkan suatu bilangan bulat : ");
      scanf("%d", &n);
      hasil = fak(n);
printf("%d! =  %d", n, hasil);
}
```

Sekarang cobalah ketikan program di atas dan masukkan beberapa bilangan untuk mengujinya!Manakah yang memiliki kecepatan yang lebih baik: pendekatan rekursif atau pendekatan iteratif?

Contoh #2: Fibonacci

Fungsi rekursif juga dapat digunakan untuk menyelesaikan bilangan Fibonacci. Fungsi Fibonacci dapat dinyatakan dalam bentuk rekursif seperti berikut:

fib(n) = 0, untuk n = 0

fib(n) = 1, untuk n = 1

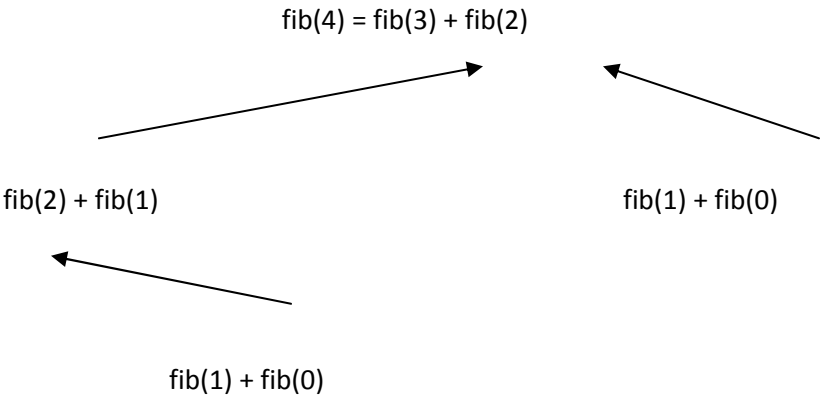
fib(n) = fib(n-1) + fib(n-2), untuk n > 1

Modul I

Berikut contoh hubungan antara n dan hasil fungsi:

n	fib(n)
0	0
1	1
2	1
3	2
4	3
5	5
6	8
…	…

Berikut contoh proses pemanggilan fib(4)



Modul I

Implementasi program dengan metode iteratif

```
#include <stdio.h>

int fib(int n)
{
     int f1 = 0, f2 = 1, fibo;
     if(n == 0)
         return 0;
     else if(n == 1)
            return 1;
     else
     {
           for(int i = 0;i < n;i++)
          {
                fibo = f1 + f2;
                f2 = f1;
                f1 = fibo;
           }
     return fibo;
     }
  }

void main()
{
        int n, hasil;
        printf("Bilangan Fibonacci ke-");
        scanf("%d", &n);
        hasil = fib(n);
```

```
        printf("fib(%d) =  %d", n, hasil);
}
```

Implementasi program dengan metode rekursif.

```
#include <stdio.h>

  int fib(int n)
  {
       if(n == 0)
            return 0;
       else if(n == 1)
            return 1;
            else
                 return fib(n-1) + fib(n-2);
  }
  void main()
  {
       int n, hasil;
       printf("Bilangan Fibonacci ke-");
       scanf("%d", &n);
       hasil = fib(n);
printf("fib(%d) =  %d", n, hasil);
  }
```

Cobalah ketikkan implementasi program dari metode rekursif di atas dan jalankan program tersebut!Uji dengan memasukkan beberapa bilangan!Selanjutnya, kembangkan program tersebut sehingga dapat untuk menampilkan n buah bilangan fibonacci!

Contoh #3: Menara Hanoi

Modul I

Menara Hanoi adalah persoalan untuk memindahkan tumpukan piring dari suatu tonggak ke tonggak lain dengan bantuan sebuah tonggak perantara. Penyelesaian secara rekursif untuk persoalan ini untuk n buah piring:

1. Pindahkan n-1 piring teratas pada tonggak A ke tonggak B, dengan menggunakan tonggak C sebagai perantara.
2. Pindahkan 1 piring tersisa pada tonggak A ke tonggak C.
3. Pindahkan n-1 piring teratas pada tonggak B ke tongak C, dengan menggunakan tonggak A sebagai perantara.

Berikut implementasi program dengan metode rekursif.

```
#include <stdio.h>
void tonggak(int n, char a, char b, char c)
{
      if(n == 1)
        printf("Pindahkan piring dari %c ke %c\n", a, c);
      else
     {
            tonggak(n-1, a, c, b);
            tonggak(1, a, b, c);
            tonggak(n-1, b, a, c);
     }
}
void main()
{
      int jml_piring;
      printf("Jumlah piringan: ");
      scanf("%d", &jml_piring);
      tonggak(jml_piring, 'A', 'B', 'C');
}
```

Latihan

Modul I

Berikut ini diberikan contoh program untuk menghitung pangkat sebuah bilangan dengan menggunakan metode iteratif.

```
#include <stdio.h>


int pangkat (int a,int b)
{
     int i, bil = a;
     if(b==1)
            return a;
     else
     {   for (i=2;i<=b;i++)
          a = a * bil;
          return a;
     }
}
void main()
{     int x,y,hasil;
        printf("masukan bilangan:");
        scanf("%i",&x);
        printf("masukan pangkat:");
        scanf("%i",&y);
        hasil = pangkat (x,y);
printf("%i",hasil);
}
```

Ketikkan program di atas dan jalankan!Ujilah dengan memasukkan beberapa bilangan. Kemudian ubahlah program tersebut sehingga program tersebut menggunakan metode rekursif!

Soal

Modul I

1. Buatlah program untuk menghitung FPB(faktor persekutuan terbesar) dari 2 bilangan dengan menggunakan metode rekursif! Untuk membantu, berikut diberikan penyelesaian secara rekursif dari FPB :

   fpb(x,y) = y jika y <= x dan sisa_pembagian(x,y) = 0

   fpb(x,y) = fpb(y,x) jika x < y

   fpb(x,y) = fpb(y, sisa_pembagian(x,y)) untuk keadaan yang lain

2. Buatlah suatu fungsi untuk membalik suatu bilangan dengan cara rekursif. Sebagai contoh, bilangan 1238 ditampilkan menjadi 8321!

POINTER

Pointer adalah suatu variabel penunjuk yang menunjuk pada suatu alamat memori komputer tertentu. Pointer merupakan variabel level rendah yang dapat digunakan untuk menunjuk nilai integer, character, float, double, atau single, dan bahkan tipe-tipe data lain yang didukung oleh bahasa C. Variabel biasa, sifatnya statis dan sudah pasti, sedangkan pada pointer sifatnya dinamis dan dapat lebih fleksibel. Variabel pointer yang tidak menunjuk pada nilai apapun berarti memiliki nilai NULL, dan disebut sebagai dangling pointer karena nilainya tidak diinisialisasi dan tidak dapat diprediksi.

Pendeklarasian variabel pointer menggunakan tanda * sebelum nama variabelnya, sedangkan untuk menampilkan nilai yang ditunjuk oleh suatu variabel pointer, juga digunakan operator * (tanda asterisk). Jika diinginkan untuk menampilkan alamat tempat penyimpanan nilai yang ditunjuk oleh suatu variabel pointer, digunakan operator & (tanda ampersand).

Pada suatu tipe data array, variabel pointer hanya perlu menunjuk pada nama variabel arraynya saja tanpa perlu menggunakan tanda ampersand, atau menunjuk pada nama variabel array pada indeks yang ke nol nya. Untuk lebih jelasnya, silahkan lihat contoh berikut:

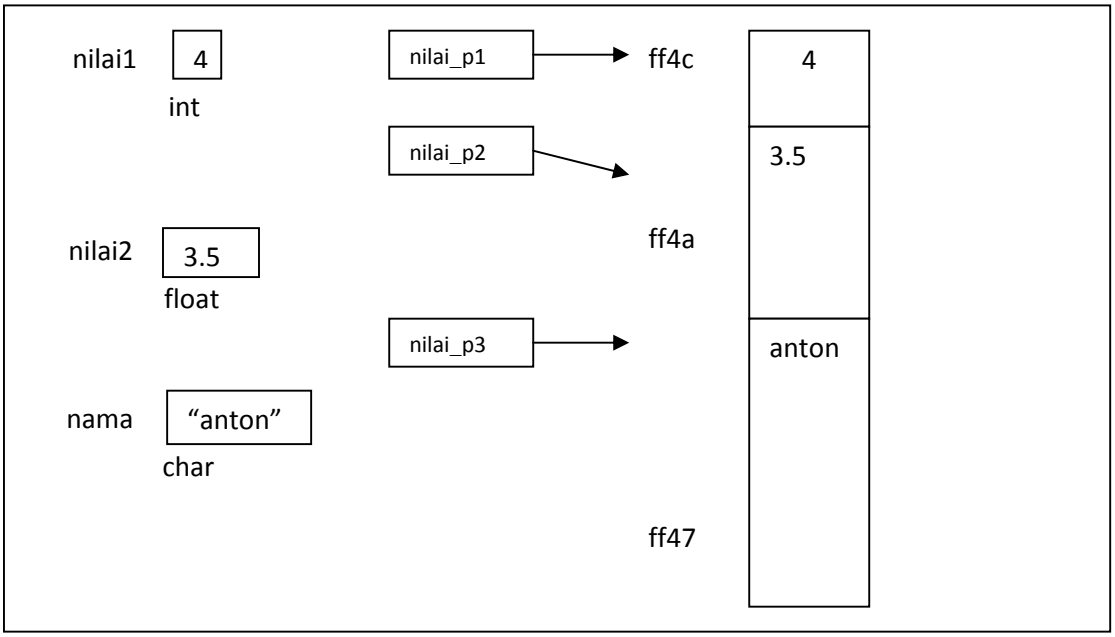
Pendeklarasian variabel biasa dan pointer:

```
//variabel biasa
int nilai1 = 4;
```

Modul I

```
float nilai2 = 3.5;
char nama[10] = “anton”; //array of char (string)

//variabel pointer
int *nilai_p1; //dangling pointer
int *nilai_p2 = &nilai1; //menunjuk ke tipe data int
char *nilai_p3 = nama; //menunjuk ke tipe data array of char
char *nilai_p4 = &nama[0];
```

Ilustrasi Pointer:



Contoh program untuk dicoba:

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

int main(){
    int nilai1 = 4;
    int nilai2 = 5;
    float nilai3 = 3.5;
```

Modul I

```
    char nama[11] = "abcdefghij";


    int *nilai_p1 = &nilai1;
    int *nilai_p2 = &nilai2;
    char *nilai_p4 = nama;
    float *nilai_p3= &nilai3;




    printf("nilai 1 = %d, alamat1 = %p, ukuran
%d\n",*nilai_p1,&nilai_p1,sizeof(nilai1));
    printf("nilai 2 = %d, alamat2 = %p, ukuran
%d\n",*nilai_p2,&nilai_p2,sizeof(nilai2));
    printf("nilai 3 = %f, alamat3 = %p, ukuran
%d\n",*nilai_p3,&nilai_p3,sizeof(nilai3));
    printf("nilai 4 = %s, alamat4 = %p, ukuran
%d\n",nama,&nilai_p4,sizeof(nama));
    getch();
    return 0;
}
```

Operasi pointer:

1. Operasi Pemberian nilai
   Contoh 1:

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


int main(){
    float nilai,*p1,*p2;
      nilai = 14.54;
      printf("nilai = %2.2f, alamatnya %p\n",nilai,&nilai);
      p1 = &nilai;
      printf("nilai p1 = %2.2f, p1 menunjuk alamat %p\n",*p1,p1);
      //pada awalnya p2 masih dangling pointer
```

```
      printf("mula-mula nilai p2 = %2.2f, p2 menunjuk alamat
%p\n",*p2,p2);

      p2 = p1;    //operasi pemberian nilai, berarti alamat x2 sama
dengan x1

      printf("sekarang nilai p2 = %2.2f, p2 menunjuk alamat
%p\n",*p2,p2);

      getch();

}
```

Contoh 2:

```
#include <stdio.h>

#include <conio.h>


int main(){

    int *p,a=25,b;

    //p masih dangling

    printf("nilai a = %d di alamat a = %p,\n",a,&a);

    printf("nilai p di alamat = %p\n",p);

    p = &a;

    printf("nilai p = %d di alamat %p\n",*p,p);

      //b diisi dengan nilai yang berasal dari nilai

    //variabel a yang ditunjuk oleh pointer p

    b = *p;

      printf("nilai b = %d di alamat %p\n",b,&b);

    getch();

}
```

Contoh 3:

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


int main(){
    int a=25,b=12,t;
      int *p,*q;
      p = &a;
      q = &b;
      printf("nilai yang ditunjuk p = %d di alamat %p\n",*p,p);
      printf("nilai yang ditunjuk q = %d di alamat %p\n",*q,q);
      //Contoh kasus, penukaran nilai 2 variabel dengan pointer
      t = *p;
    *p = *q;
    *q = t;
      printf("nilai yang ditunjuk p sekarang = %d di alamat
%p\n",*p,p);
      printf("nilai yang ditunjuk q sekarang = %d di alamat
%p\n",*q,q);
    getch();
}
```

Contoh 4:

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


int main(){
    int a,*p;
      p=&a;
      *p=25;
      printf("nilai a = %d",a);
}
```

2. Operasi Aritmatika

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

Modul I

```
#include <conio.h>


int main(){
    int a,b=10,*p,*q;
      p=&a;
      *p=25;
      printf("nilai a = %d\n",a);
      printf("alamat p = %p\n",p);
      q=&b;
      printf("alamat q = %p\n",q);
      printf("nilai a + b = %d\n",(*p+*q));
    //posisi alamat p menjadi bergeser, nilai berubah
    p=p+1;
    printf("nilai p = %d, alamat = %p\n",*p,&p);
    q=q-1;
    printf("nilai q = %d, alamat = %p\n",*q,&q);
      getch();
}
```

Pointer pada array 1 dimensi:

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


int main(){
char S[] = "anton";
      char *p;
      //cara 1
      //langsung menunjuk nama array.
      p=S;
      for(int i=0;i<5;i++){
            printf("%c",*p);
            p++;
```

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

	//cara 2
	p=&S[0];
for(int i=0;i<5;i++){
		printf("%c",*p);
		p++;
	}
	printf("\n");

	//Membalik kalimat
	p--;
	for(int i=0;i<5;i++){
		printf("%c",*p);
		p--;
	}
	printf("\n");

	getch();
}
```

Pointer pada Struct:

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

typedef struct{
	int nim;
	int umur;
	float ipk;
} Mahasiswa;
```

Modul I

```
Mahasiswa m;
Mahasiswa *p = &m;

int main(){
    //struct biasa
    m.nim=123;
    m.ipk=3.2;
    m.umur=23;
    printf("nim = %d\n",m.nim);
    printf("ipk = %f\n",m.ipk);
    printf("umur = %d\n",m.umur);

    //struct pointer
    p->ipk = 3.5;
    p->nim = 321;
    p->umur = 32;
    printf("nim = %d\n",p->nim);
    printf("ipk = %f\n",p->ipk);
    printf("umur = %d\n",p->umur);

    //mengacu pada variabel aslinya
    printf("nim = %d\n",m.nim);
    printf("ipk = %f\n",m.ipk);
    printf("umur = %d\n",m.umur);
      getch();
}
```

Pengembangan:

Buatlah sebuah program untuk mengecek apakah suatu kata palindrom atau bukan, tanpa memperhatikan spasi dan huruf besar/kecilnya. Program dibuat dengan menggunakan template struct sebagai berikut:

```
typedef struct{
```

```
        char elemen[30];
        int jml_kata;
} Kata;

Kata kata;
Kata *p_kata=&kata;
```

Lanjutkanlah program berikut agar hasilnya sesuai dengan soal di atas:

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

typedef struct{
        char elemen[30];
        int jml_kata;
} Kata;

Kata kata;
Kata *p_kata=&kata;

int main(){
    char kalimat[30];
    p_kata->jml_kata=0;
    char *p = p_kata->elemen;
    printf("Masukkan kata : ");gets(kalimat);
    fflush(stdin);
    printf("Kalimat : %s\n",kalimat);
    for(int i=0;i<kalimat[i];i++){
      *p=kalimat[i];
      p_kata->jml_kata++;
      p++;
    }
```

```
    p=p_kata->elemen;
    //tampilkan kembali kalimat tersebut
    for(int i=0;i<=p_kata->jml_kata;i++){
      printf("%c ",*p);
      p++;
    }

    //kembangkan….

getch();
}
```

Soal:

Buatlah program data KTP, dengan menggunakan pointer pada struct KTP sebagai berikut:

```
typedef struct
{
      int tgl;
      int bln;
      int th;
}Tanggal;

typedef struct
{
      char noID[5];
      char nama[30];
      char jenis_kelamin; //'L' atau 'P'
      Tanggal t;
}KTP;

typedef struct
{
```

```
      KTP ktp;
      int jml;
}Data_KTP;

Data_KTP data_ktp;
Data_KTP *p;
```

Buatlah fungsi untuk:

1. Menambah data
2. Mencari data berdasarkan tahun kelahiran tertentu
3. Menampilkan data berdasarkan L dan P
4. Mengedit data

Semua pengaksesan dilakukan dengann menggunakan pointer.

STRUCT

- Bentuk struktur data yang dapat menyimpan variabel-variabel dalam 1 nama, namun memiliki tipe data yang berbeda ataupun sama. Variable-variabel tersebut memiliki kaitan satu sama yang lain.

Bentuk umum :

```
typedef struct nama_struct{
tipe_data <nama_var>;
tipe_data <nama_var>;
....
};
```

DEKLARASI

Ada 2 cara pendeklarasian struct, yaitu :

Deklarasi 1:

```
typedef struct Mahasiswa {
char NIM[8];
char nama[50];
```

```
float ipk;
};
```

Deklarasi 2 :

```
struct {
char NIM[8];
char nama[50];
float ipk;
} mhs;
```

Contoh struct:

```
#include <iostream.h>


void main()


{
 struct orang
 {
      char nama[40];
      short umur;
 }saya;
cout<<"nama : ";
 cin.getline(saya.nama,40);
cout<<"umur :" ;
cin>>saya.umur);
cout<<saya.nama<<saya.umur;
}
```

ARRAY OF STRUCT

Apabila hendak menggunakan 1 struct untuk beberapa kali, ada 2 cara :

2. Deklarasi manual

Contoh :

```
   #include <stdio.h>
      typedef struct Mahasiswa {
         char NIM[8];
      char nama[50];
         float ipk;
      };
void main()
{
         Mahasiswa a,b,c;
   ……
         ……
         ……
}
```

artinya struct mahasiswa digunakan untuk 3 variabel, yaitu a,b,c

2. Array of struct

Contoh :

```
   #include <stdio.h>
      typedef struct Mahasiswa {
         char NIM[8];
      char nama[50];
         float ipk;
      };
void main()
{
         Mahasiswa mhs[3];
   ……
         ……
```

```
        ……
}
```

artinya struct mahasiswa dapat digunakan untuk tiga variabel mhs, yaitu mhs[0], mhs[1], dan mhs[2].

Contoh lainnya :

```
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
typedef struct orang
{
 char nama[30];
 short umur;
 };
void main()

{
 orang saya[5];
 int i,x;
 for(i=0;i<=4;i++)
 {
printf("nama ke-%i : ",i+1);
cin.getline(saya[i].nama,30);
printf("umur ke-%i : ",i+1);
scanf("%i",saya[i].umur);
printf("%s berumur %i",saya[i].nama,saya[i].umur);
 }
 for(x=0;x<=4;x++)
 {
printf("nama %s berumur %d",saya[x].nama,saya[x].umur);
 }
```

```
}
```

LATIHAN TERBIMBING STRUCT

Buatlah struct untuk buku dengan deklarasi manual.

Ketentuan :

Yang harus disimpan adalah judul buku, tahun terbit dan harga buku.

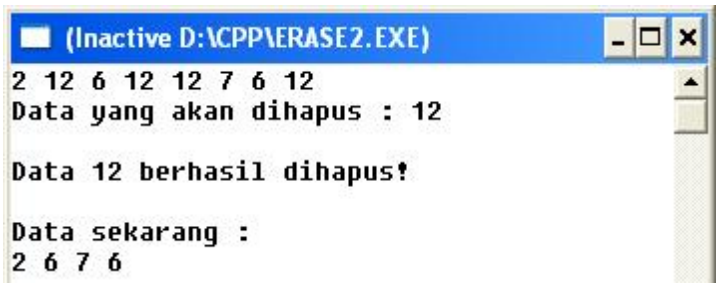
Source code :

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

typedef struct buku{
      char judul[15];
      int tahun_terbit;
      int harga;
};

void main(){

 buku book;
```

LATIHAN - LATIHAN MANDIRI DI KELAS

5. Program penghapusan data dengan inputan berupa angka yang ingin dihapus oleh user.
   Ketentuan :

- Semua data yang sesuai dengan inputan user akan terhapus.
- Bonus jika terdapat counter untuk menghitung berapa data yang terhapus.
- Capture :



6. Program untuk melakukan update data.
   Ketentuan :
   - Terdapat dua inputan, yaitu inputan data yang akan diubah dan data baru (data pengganti)
   - Semua data yang sesuai dengan inputan user akan diupdate nilainya.

7. Program untuk melakukan penambahan data.
   Ketentuan :
   - Penambahan data dapat dilakukan di mana saja.
   - Inputan dari user berupa :
     - Nilai yang akan ditambahkan
     - Indeks ke berapa yang dituju
   - Setelah penambahan, maka jumlah data akan bertambah dan posisi data akan bergeser sesuai dengan penambahan yang telah dilakukan.

8. Buatlah struct untuk data lagu yang berisi tentang judul lagu, penyanyi, tahun produksi, nomor track dan kode album.
   Ketentuan :
   - program ini akan memiliki dua buah struct, yaitu struct lagu dan struct kodeRBT.
   - Jumlah data yang diinputkan dinamis (maks. 20 lagu)

LATIHAN – LATIHAN MANDIRI DI RUMAH / TES

3. Buatlah menu add, edit, view dan delete data menggunakan array.

Note : operasi-operasi tersebut dapat dilakukan pada data dan indeks mana saja. (Inputan data dan indeks dinamis).

4. Buatlah dengan menggunakan struct dan array 1 dimensi : record peminjaman buku di perpustakaan.
   Data yang akan ditampilkan sebagai output adalah :

   - Nama
   - NIM
   - Tanggal peminjaman (dd/mm/yyyy)
   - Kode buku, dengan format **nomor rak-kategori buku** :
     - Nomor rak (inputan terserah)
     - Kategori : R (Referensi) atau U (Umum)
     - Contoh : 1234-R

   Gunakan tiga struct untuk kasus ini!

Gabungkan soal 1 dan 2 di atas sehingga menghasilkan program berisi struct yang dapat melakukan fungsi add, edit, view dan delete

Modul I

Contoh tampilan program :

```
Berapa data yang anda inputkan ? 1
Data ke-1:
      Judul lagu : Bersamamu
      Penyanyi : Vierra
      Tahun Produksi : 2009
      Nomor Track : 3
      Kode Album : MFL

Data lagu yang anda miliki :
   Data 1:
      Penyanyi : Vierra
      Judul lagu : Bersamamu
      Kode RBT : 3-MFL
      Tahun : 2009
```

<Asisten-asisten dapat memodifikasi atau mengembangkan soal-soal di atas sesuai dengan kelasnya>

END

SENARAI BERANTAI TUNGGAL TIDAK BERPUTAR (SINGLE LINKED LIST NON CIRCULAR)

DEKLARASI LINKED LIST

Sebelum membuat sebuah senarai (Link List) kita harus mendeklarasikan tipe data dan pointer yang akan kita gunakan. Selain itu kita juga harus mendeklarasikan list-list yang nantinya akan kita gunakan. Sebagai contoh:

```
#include <iostream.h>
typedef struct Gerbong        //membuat sebuah tipe data baru yang terdiri dari
2 field
{
      int data;          //field data bertipe integer
      Gerbong *next;           //field next merupakan pointer dari list
};
Gerbong *baru;                 //variable baru bertipe pointer dari list
Gerbong *kepala;         //variable kepala bertipe pointer dari list
Gerbong *ekor;                 //variable ekor bertipe pointer dari list
Gerbong *bantu;          //variable bantu bertipe pointer dari list
Gerbong *hapus;          //variable hapus bertipe pointer dari list
```

catatan:

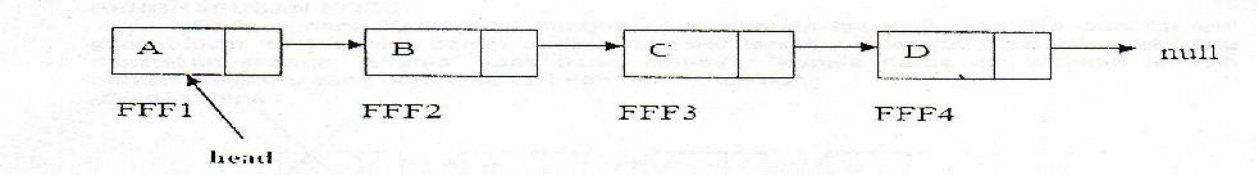
- 'Gerbong' diatas merupakan nama dari tipe data bentukan.
- kepala, ekor, baru, bantu,hapus merupakan variable pointer yang berisi alamat memori.
- kepala merupakan node paling awal dan selalu tetap(tidak boleh berpindah).
- ekor merupakan node paling akhir jadi ketika terjadi penambahan sebuah node yang diletakan di belakang maka pointer dari ekor akan ikut berpindah.
- baru merupakan nama untuk node yang diciptakan atau ditambahkan.
- bantu, hapus digunakan untuk memudahkan operasi edit dan delete sebuah node.

ILUSTRASI

Senarai berantai dapat diilustrasikan seperti satu kesatuan rangkaian kereta api. Kereta api terdiri dari beberapa gerbong, masing-masing dari gerbong itulah yang disebut struct / tipe data bentukan. Agar gerbong-gerbong tersebut dapat saling bertaut dibutuhkan minimal sebuah kait yang dinamakan sebagai pointer.

Setelah mendeklarasikan tipe data dan pointer pada list, selanjutnya kita akan mencoba membuat senarai / linked list tunggal tidak berputar atau sebuah gerbong. Ada beberapa operasi yang dapat kita buat pada senarai tersebut, diantaranya: tambah, hapus dan edit dari gerbong tersebut.

Inti dari linked list adalah proses (tambah, edit, hapus) dari gerbong / node dan bagaimana rnenyambungkan antar gerbong / node tersebut.



Gambar diatas dilustrasikan sebuah rangkaian kereta api dengan 4 buah gerbong. Gerbong A akan disebut sebagai kepala / head (walaupun penamaan ini bebas) dan gerbong D adalah ekor / tail. Tanda panah merupakan kait atau pointer yang menghubungkan satu gerbong dengan yang lainnya.

Pointer yang dimiliki D menuju ke NULL,inilah yang membedakan antara senarai berputar dengan yang tidak berputar. Kalau senarai berputar maka pointer dari D akan menuju ke A lagi.

PEMBENTUKAN NODE (GERBONG)

Digunakan keyword new yang berarti mempersiapkan sebuah node baru berserta alokasi memorinya, kemudian node tersebut diisi data dan pointer nextnya ditunjuk ke NULL. Pembentukan node tidak dapat dilakukan sekaligus namun harus satu persatu, hal ini berkaitan dengan bagaimana cara menyambungkan antar node tersebut.

Modul I

```
kepala= new Gerbong;     //membuat sebuah node dengan nama kepala
kepala->next==NULL;      //pointer milik kepala diarahkan ke NULL
kepala->data=101;        //isi field data dengan 101
```

Kode diatas apabila dicompile maka akan membentuk sebuah node bertipe Gerbong dengan nama kepala. Karena bertipe list maka kepala juga memiliki 2 field seperti list yaitu field data dan field next. Bila diasumsikan dengan gerbong maka pada proses ini dibentuk gerbong A.

Setelah membentuk sebuah node dengan nama kepala, maka langkah selanjutnya adalah membentuk gerbong yang lain (gerbong B dan selanjutnya).

```
baru=new Gerbong;              //membuat sebuah node dengan nama baru
baru->data=5;
```

//Tambah Depan

```
if (kepala->next==NULL)
{
      baru-->next=kepala;           //rnenyambung node bare dengan node kepala
      kepala=baru;                  //memindahkan kepala ke node baru
}
```

//Tambah Belakang

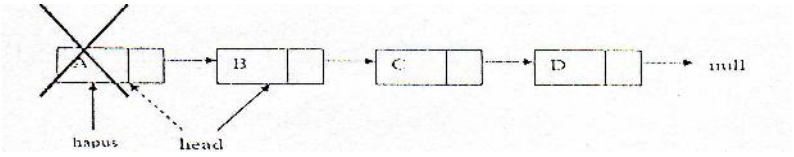
```
ekor->next=baru;           //menyambung node paling akhir/ekor ke node
baru
ekor=baru;             //meniindahkan ekor ke node terakhir yaitu node baru
.
ekor->next=NULL;
```

Tambah belakang berarti node baru yang terbentuk akan diletakan di posisi paling belakang.

PENGHAPUSAN NODE

Pada senarai berkepala, penghapusan sebuah list dilakukan jika ada list lain yang bukan list "kepala" dalam barisan senarai tersebut. Dengan kata lain, list yang digunakan sebagai "kepala" tidak boleh dihapus, "kepala harus dipindahkan terlebih dahulu. Keyword yang digunakan adalah delete. Contoh:

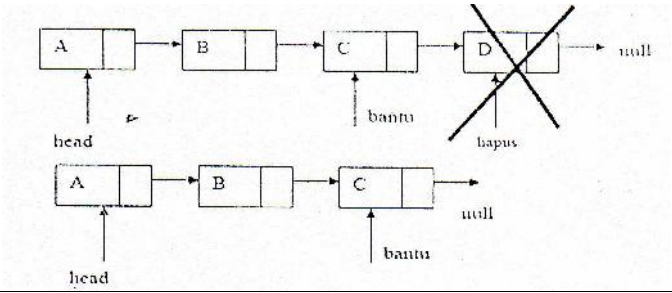
//Hapus Depan




```
if(kepala->next!=NULL)
{     if(kepala->next==ekor)
      {     hapus=kepala;
      kepala=kepala->next;
      delete hapus;
      }
}
else
{
```

```
      kepala=NULL;
}
```

//Hapus Belakang




```
if(head->next!=NULL)
{
      bantu=kepala;
      while(bantu->next->next!=NULL)
      {     bantu=bantu->next;
      }
      hapus=bantu->next;
      bantu->next = NULL;
      delete hapus;
}
else
{
      head =NULL;
}
```

MENAMPILKAN ISI NODE

Untuk menampilkan isi suatu node dengan cara mengakses field yang ada di dalam node tersebut. Yaitu dengan →Contoh :

```
bantu =head;
while(bantu!=NULL)
{
      printf("%i ",bantu->data);
      bantu=bantu->next;
}
```

LATIHAN

1. Buatlah sebuah linked list non circular yang berisi nim anda dan nama lengkap anda.
2. Buat fungsi untuk menambahkan node single linked list non circular dimana tiap node mengandung informasi nim dan nama. Peletakan posisi node urut berdasar nim secara ascending, jadi bisa tambah depan, belakang maupun tambah di tengah.
   Isikan data nim dan nama lengkap teman sebelah kiri dan kanan anda!!!
3. Buatlah fungsi untuk menampilkan data 3 buah node yang telah anda bentuk sebelumnya.
   Contoh tampilan
   NIM Nama Lengkap
   22053766 Hernawan

22053768 Andrew S
22053791 Anthony S

4. Buatlah fungsi untuk mencari nama yang telah diinputkan dengan menggunakan NIM.
   Contoh tampilan:
   Inputkan nim yang dicari = 22053768
   Nama yang tercantum Andrew S
5. Buatlah sebuah fungsi untuk menghapus nim yang diinputkan oleh user.
   Contoh tampilan:
   NIM yang mau dihapus = 2205376
   NIM dengan nama Andrew S ditemukan dan telah dihapus
6. Buatlah sebuah program dengan menggunakan single linked list non circular dengan fungsi-fungsi:
   - menambah data(dari depan dan dari belakang)
   - search data yang telah diinputkan
   - menghapus data(dari depan dan dari belakang)
   - mencetak data

   Buat dengan menggunakan menu ya…..
7. (soal extended) Dengan menggunakan soal no 6 tambahkan 4 fungsi tambahan:
   - menambah data(di tengah)
   - menghapus data tertentu(berada di tengah)
   - mengurutkan data acak secara acending
   - mengurutkan data acak secara decending

SELAMAT MENGERJAKAN

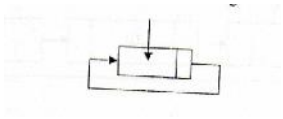
Modul I

SENARAI BERANTAI TUNGGAL BERPUTAR
(SINGLE LINKED LIST CIRCULAR)

- ✓ Perbedaan Single Linked list Circular dengan Non Circular

Pada saat kita membahas Single Linked List non Circular telah dijelaskan bahwa kita membutuhkan sebuah kait untuk menghubungkan node-node / gerbong-gerbong data yang ada. Perbedaan dengan Single Linked List Circular adalah pada node terakhir/ ekor yang semula menunjuk ke NULL diganti menunjuk kembali ke kepala atau head.

Contoh Struct Single Link List Circular



Dengan melihat gambar kita dapat membayangkan bahwa setiap node akan membentuk lingkaran, dan semakin banyak node lingkaran akan semakin besar. **Pada intinya tail akan selalu terhubung dengan head.**

Membuat Single Linked List Circular berkepala berputar

Untuk menginisialisasikan Single Linked List Circular berkepala berputar cukup mudah karena kita hanya seperti memberikan label pada suatu list. Cara mendeklarasikanna seperti contoh dibawah ini:

```
#include “stdio.h”
typedef struct Tnode
{  int data;
   Tnode *next;
};
Tnode *bantu;        //mendeklarasikan pointer bantu
Tnode *baru;         //mendeklarasikan pointer baru
Tnode *head;         //mendeklarasikan pointer head
Tnode *tail;         //mendeklarasikan pointer tail
void main()
{
   head=new Tnode;
   head ->next=head;
}
```

//memasukkan data Single Linked list Circular berkepala berputar

```
head -> data = 50;
```

//menambah simpul baru dan menghubungkannya dengan simpul kepala

```
baru = new Tnode;
baru -> data = 20;              apakah perintah ini masih dapat digunakan
untuk penambahan
baru -> next = head;            simpul berikutnya???
head ->next = head;
```

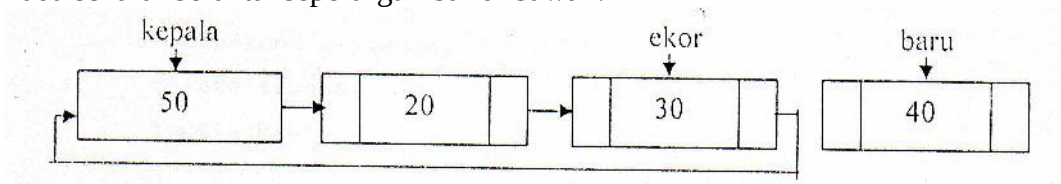
//menambah simpul baru dan membuat ekor pada akhir senarai

```
tail = baru;
```

Modul I

```
baru = new Tnode;
baru -> data = 30;                apakah perintah ini masih dapat digunakan
untuk penambahan
baru ->next=head;                 simpul berikutnya???
tail -> next=baru;
tail = baru;
```

✓ Pada senarai berantai seperti gambar di bawah:



✓ Bagaimana cara meletakkan data baru 40 diantara simpul dengan data 50 dan 20??
Tambahkan pada program diatas:

```
baru=new Tnode;
baru->data=40;
baru->next=head->next;
head->next=baru;
```

✓ Mengedit / update data
Misalnya kita ingin mengganti data pada ekor menjadi 25, caranya adalah sebagai berikut:

```
tail->data=25;
```

✓ Membaca senarai
Tambahkan pada program diatas

```
bantu=head;
printf("%d", bantu);
bantu=bantu->next;
while(bantu!=head)
{
      printf("%d", bantu);
bantu=bantu->next;
}
```

✓ Menghapus Simpul ekor
Tambahkan pada program diatas:

```
bantu=head;
while(bantu->next!=tail)
{
   bantu=bantu->next;
}
bantu->next=head;
delete tail;
tail=bantu;
```

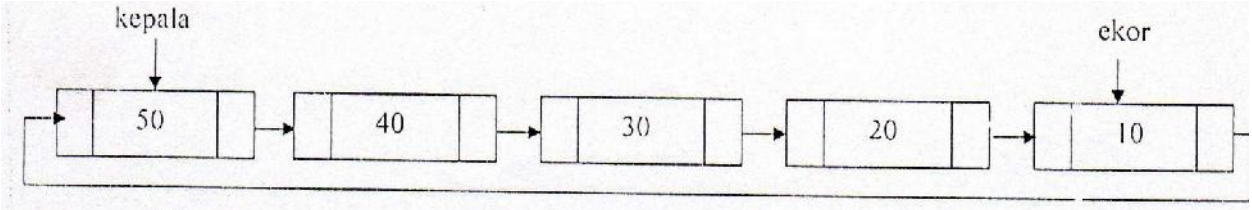
✓ Menghapus Simpul kepala/head
Tambahkan pada program diatas:

```
bantu=head->next;
delete head;
head = bantu;
```

Modul I

Latihan

1. Buatlah sebuah Single Linked List Circular yang mcmpunyai 5 buah simpul seperti pada contoh dibawah ini



   a. Tuliskan listing program untuk membaca data dari seluruh simpul dengan menggunakan perulangan (for/while)
   b. TuIiskan listing program untuk menambah sebuah simpul baru dengan data 5 yang diletakan diantara data 30 dan data 20.
   c. Tuliskan listing program untuk menghapus simpul dengan data 20 tanpa merusak senarai.
   d. Tuliskan listing program untuk menghapus semua simpul dengan menggunakan variabel bantu dengan menggunakan perulangan (while/for).

2. Buatlah program Single Linked List Circular yang fleksibel dan dinamis. Dimana user dapat menambah simpul baru beserta datanya, membaca semua data pada senarai, menghapus simpul dengan data yang ditentukan oleh user, dan dapat menghapus semua simpul yang ada. Jangan Iupa membuat trapping errornya.
3. (soal extended) Buatlah sebuah link list berisi data-data identitas ktp(nama, alamat, no KTP) buatlah menu sebagai berikut:
   a. Input entry ktp (input akan langsung terurutkan berdasarkan no ktp asending)
   b. Delete file yang diinginkan user.
   c. Searching entry ktp
   d. Cetak semua entry ktp
   e. Cetak entry ktp tertentu
4. (soal extended) Buatlah 3 link list yang merepresentasikan jam(1-12), menit(1-60) dan detik(1-60). Mintalah inputan dari user berupa jumlah detik. Konversikan jumlah detik tersebut dan masukkan ke dalam link list (gunakan pointer jam,menit,detik) kemudian cetak jam yang tertera di link list tersebut.