PRORAM STUDI TEKNIK INFROMATIKA-D3
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

Modul Praktikum

ALGORITMA DAN STRUKTUR DATA

Karis Widyatmoko,S.Si, M.Kom
Fikri Budiman, M.Kom

PRORAM STUDI TEKNIK INFROMATIKA-D3
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

Modul Praktikum

ALGORITMA DAN STRUKTUR DATA

Karis Widyatmoko,S.Si, M.Kom
Fikri Budiman, M.Kom

Modul Praktikum Algoritma dan Struktur DataI

Tim Penyusun Karis Widyatmoko,S.Si, M.Kom
Fikri Budiman, M.Kom

Pengulas Tim Pengampu

Edisi Kedua

April 2017

Program Studi Teknik Informatika-D3

Fakultas Ilmu Komputer

Universitas Dian Nuswantoro

LEMBAR PENETAPAN

Menetapkan bahwa Modul Praktikum Algoritma dan Struktur Data mulai digunakan di semester genap Tahun Akademik 2016/2017 di lingkungan Program Studi Teknik Informatika-D3, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro.

Semarang, 7 April 2017

Ketua Program Studi TI-D3

Muslih, M.Kom
NPP: 0686.11.1996.082

VISI DAN MISI
PROGRAM STUDI TEKNIK INFORMATIKA-D3

Visi

"Menjadi Pilihan Utama Dalam Pendidikan Vokasi Bidang Teknologi Informasi, dan Mampu Menghasilkan Lulusan Dengan Keahlian Teknologi Web dan Mobile yang Cakap Dalam Bidang *Technopreneurship*"

Misi

a. Tuntutan dosen profesional yang bermutu dan Ahli dalam bidang kompetensi yang diajarkan, dan pengembangkan riset sesuai dengan kebutuhan keilmuan pada program studi.
b. Menyelenggarakan pendidikan berbasis teknologi informasi yang profesional dan bermutu yang cakap dalam bidang technopreuner, dan meningkatkan kompetensi mahasiswa dengan sertifikasi uji kompetensi pada bidang web dan mobile programming.
c. Menyelengarakan kerjasama dengan berbagai lembaga pihak luar, sehingga dapat mengontrol penyelengaraan pendidikan yang selalu mutakhir dan dapat diterapkan secara tepat guna.

PROFIL LULUSAN

PROGRAM STUDI TEKNIK INFORMATIKA-D3

Profil	Deskripsi Profile
Programmer	Dapat mengimplemetasikan analisis problem, obyek serta spesifikasi permasalahan hingga tertuang dalam logical flow maupun rancangan model sehingga dapat diselesaikan dengan bahasa pemrograman yang dikuasai khususnya web dan mobil aplikasi dengan penerapan multimedia.
Network Engineer	Dapat Mengelola perangkat-perangkat networking, melakukan monitoring, troubleshooting serta mengaplikasikan keamanan jaringan agar berjalan dengan baik dari sebuah perusahaan maupun instansi.
Database Administrator	Merancang, mengelola serta mengimplementasikan database secara fisik dan mengontrol security, integritas database yang ditanganinya.
Web Developer	Sebagai profesional dari desain sampai dengan implementasi perankat lunak dengan platform webbased aplication.
Technopreneur	Sebagai pelaku usaha dalam pemanfaatan teknologi yang sedang berkembang pesat khususnya di dunia IT

TATA TERTIB PENGGUNAAN LABORATORIUM KOMPUTER UNIVERSITAS DIAN NUSWANTORO

1. Pengguna Laboratorium komputer adalah orang yang tercatat resmi sebagai mahasiswa UDINUS, kecuali peserta pelatihan atau semacamnya.
2. Pengguna Laboratorium komputer harus berperilaku sopan dan menggunakan pakaian sopan dan rapi sesuai ketentuan UNIVERSITAS (tidak memakai kaos oblong atau sandal).
3. Mahasiswa wajib mempunyai, menggunakan dan menjaga keamanan user login milik sendiri dengan rutin mengganti password secara periodik.
4. User login mahasiswa terkait dengan quota space drive yang ada di server lab komputer.
5. Tidak diperbolehkan membawa makanan, minuman kedalam laboratorium komputer dan harus membuang sampah pada tempatnya.
6. Semua barang yang ada di lingkungan laboratorium komputer merupakan barang milik Universitas Dian Nuswantoro.
7. Penggunaan perangkat tambahan yang tidak permanen terpasang di laboratorium komputer harus seijin laboran lab yang digunakan.
8. Tidak diperbolehkan merubah konfigurasi, melepas rangkaian, mencorat-coret, dan merusak barang-barang di lingkungan laboratorium komputer.
9. Tidak diperbolehkan membawa keluar barang-barang di lingkungan laboratorium komputer tanpa ijin.
10. Barang-barang yang dibawa dari luar untuk ditempatkan di lab. Komputer harus seijin Ka. Lab untuk didata keperluan dan peruntukannya.

11. Penggunaan lab diluar jam kuliah atau untuk keperluan lain selain perkuliahan harus seijin Ka Laboratorium, yang ketentuannya akan diatur kemudian.
12. Pengguna Lab. Wajib melaporkan kejadian yang membayakan orang lain di lingkungan laboratorium pada pihak yang berwenang (laboran/Satpam).

Semarang, 9 Maret 2015

Ka. UPT. Laboratorium Komputer

Elkaf Rahmawan Pramudya, M. Kom
NPP : 0686.11.1998.147

PERATURAN PRAKTIKUM

1. Mahasiswa wajib membawa modul praktikum.
2. Mahasiswa wajib hadir minimal 75% dari seluruh pertemuan praktikum di lab. Mahasiswa dilarang membuka program aplikasi yang tidak berhubungan dengan praktikum.
3. Durasi kegiatan praktikum adalah 200 menit per pertemuan.
4. Toleransi keterlambatan adalah kurang dari 30 menit.

PENILAIAN PRAKTIKUM

Penilaian praktikum dilakukan berdasarkan pemenuhan tugas yang meliputi logika pemrograman menggunakan *algoritma* atau *spoudo code*, penulisan *statement bahasa pemrograman*, evaluasi atau pengujian hasil dan penilaian ditentukan oleh dosen pengampu.

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa modul Algoritma dan Struktur data sebagai pendukung praktikum telah dapat terselesaikan.

Penyusun berharap bahwa modul ini dapat membantu mahasiswa dalam menyelesaikan mata kuliah serta mendukung terwujudnya capaian pembelajaran yang telah ditentukan.

Penyusun mengucapkan banyak terimakasih kepada semua pihak yang telah membantu penyelesaian modul ini.

Semarang, April 2017

PRAKTIKUM 1
PENGENALAN STRUKTUR DATA
BAHASA C++

Bagaimana cara membentuk sebuah program dengan konsep struktur data dari atomik sampai terstruktur yang dimplementasikan dalam kasus sederhana ?

1. Memahami masalah secara menyeluruh dan persiapan data
2. Keputusan operasi-operasi yang dilakukan terhadap data
3. Penyimpanan data-data pada memori sehingga tersimpan dan terstruktur secara logis, operasinya efisien
4. Memahami bahasa pemrograman C++ untuk jenis data yang ada

Struktur data adalah cara penyimpanan dan pengorganisasian data-data pada memori komputer maupun file secara efektif sehingga dapat digunakan secara efisien, termasuk operasi-operasi di dalamnya.

Di dalam struktur data berhubungan dengan 2 aktivitas:

1. Mendeskripsikan kumpulan obyek data yang sah sesuai dengan tipe data yang ada
2. Menunjukkan mekanisme kerja operasi-operasinya

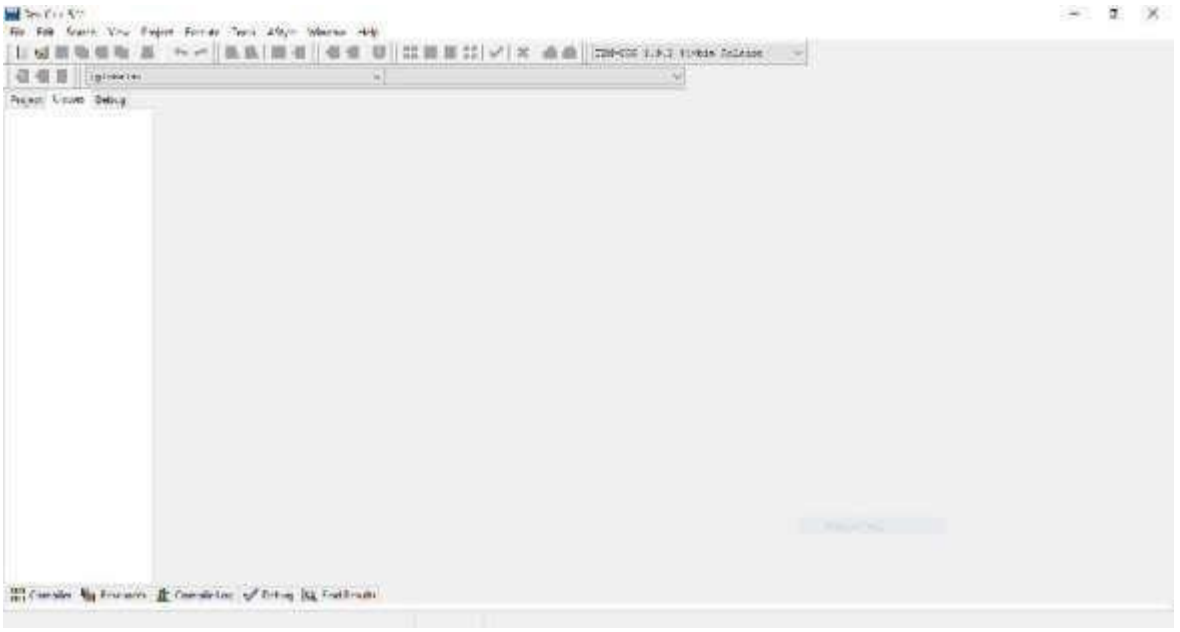
Untuk memulai membuat sebuah program, editor yang kita gunakan adalah Dev C++. Dev C++ adalah sebuah IDE (Integrated Development Environment) C/C++ yang sudah dilengkapi dengan TDM-GCC Compiler (bagian dari *GNU Compiler Collection* / *GCC*). Dev-C++merupakan IDE gratis dan full featur yang didistribusikan dibawah lisensi *GNU General Public License* untuk pemrograman C dan C++. IDE sendiri adalah Lembar kerja terpadu untuk pengembangan program.

Fungsi IDE DevC++ :

1. Menulis Program / Source Code.
2. Mengkompilasi Program (Compile)
3. Melakukan Pengujian Program (Debugging)
4. Mengaitkan Object dan Library ke Program (Linking)
5. Menjalankan Program (Running)

Langkah-langkah Praktikum

1. Buka editor **DevC++**.



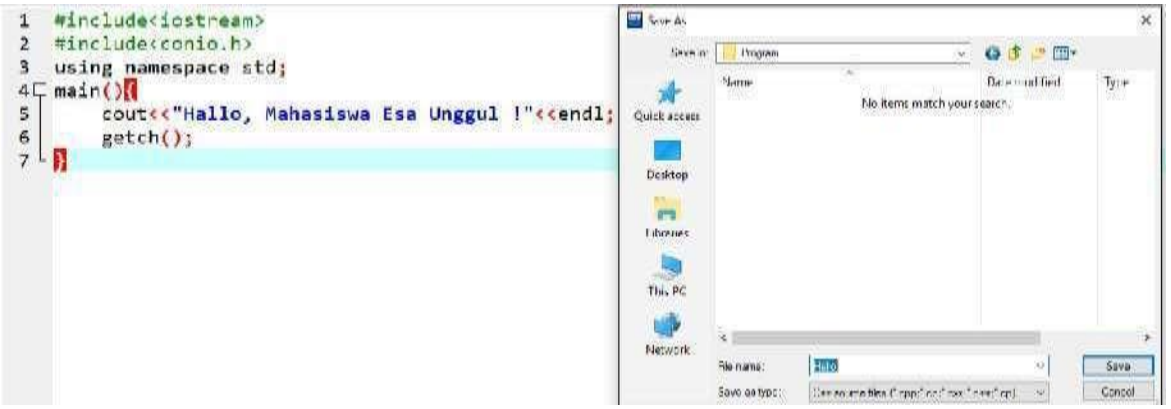
2. Buatlah file baru dengan membuka menu **File > New > Source File** atau dengan **Ctrl + N**.

3. Tulislah Program 1.1 berikut ini.

```
1  #include<iostream>
2  #include<conio.h>
3  using namespace std;
4  main(){
5      cout<<"Hallo, Mahasiswa Esa Unggul !"<<endl;
6      getch();
7  }
```

Program 1.1 Hello.cpp

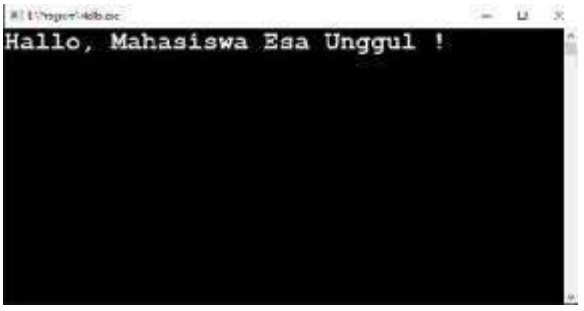
4. Simpan program yang telah dituliskan dengan membuka menu **File > Save as… .**
   Pilih lokasi penyimpanan dan beri nama file dengan "**Hello.cpp**".



5. Lakukan kompilasi program melalui menu **Execute** > **Compile** atau dengan menekan**F9**,seperti Gambar 1.1.
6. Perhatikan hasil kompilasi program pada bagian "Compile Log". Jika kompilasi sukses, maka akan ditampilkan pesan kurang lebih seperti Gambar 1.2.
7. Jalankan program (running) melalui menu **Execute** > **Run** atau dengan menekan**F10**. Hasil program akan ditampilkan pada window seperti pada Gambar 1.3.
8. Atau melalui menu Compile & Run atau menekan **F11**,untuk melakukan kompilasi dan menjalankan program secara langsung, seperti Gambar dibawah ini.

Gambar 1.1

Gambar 1.2



Gambar 1.3

Note :

Sebelum mulai melakukan coding program, diingat bahwa bahasa C atau C++ bersifat "case sensitive", yang artinya huruf besar dan huruf kecil dibedakan. Maupun dalam code-code yang digunakan jangan sampai kurang sedikitpun. "Semangat "

Tuliskan dan jalankan beberapa program berikut ini dan jelaskan maksud pemrograman berikut dengan hasil running yang terjadi !

a. Program 1

```
#include<iostream>
#include<conio.h>
#include<windows.h>
using namespace std;
void gotoxy(int x, int y){
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),coord);
}
int main(){
    gotoxy(1,4);
    cout<<"Hallo Mahasiswa";
    getch();
}
```

b. Program 2

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
    float bil1,bil2,hasil;

    bil1 = 50;
    bil2 = 50;

    hasil = bil1*bil2;

    cout<<"Hasil kali dari "<<bil1<<" dan "<<bil2<<" adalah "<<hasil;
    getch();
}
```

c. **Program 3**

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
    const int sks=4,nim=201581178;
    char nama[12]="Mia Kastina",matkul[]="Struktur Data";
    float nilai1,nilai2,nilai3;
    nilai1=90;
    nilai2=80;
    nilai3=(nilai1 + nilai2)/2;
    cout<<"Nama Mahasiswa : "<<nama<<"\n";
    cout<<"NIM            : "<<nim<<"\n";
    cout<<"Mata Kuliah    : "<<matkul<<"\n";
    cout<<"SKS            : "<<sks<<"\n";
    cout<<"Nilai Teori    : "<<nilai1<<"\n";
    cout<<"Nilai Praktek  : "<<nilai2<<"\n";
    cout<<"Nilai Akhir    : "<<nilai3<<"\n";
    getch();
}
```

PRAKTIKUM 2
ARRAY

Array atau Larik merupakan Struktur Data sederhanayang dapat didefinisikan sebagai pemesanan alokasimemory sementara pada komputer.Dalam implementasinya, Array merupakankumpulan dari nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama. Nilai-nilai data pada suatu larik disebut dengan elekmen-elemen larik. Letak urutan dari suatu larik ditunjukkan oleh suatu subscript atau index.

Karakteristik Array :

a. Mempunyai batasan dari pemesanan alokasi memory(bersifat statis)
b. Mempunyai type data sama (bersifat homogen)
c. Dapat diakses secara acak

Hal yang harus diketahui dalam mendeklarasikan array :

a. Type data array
b. Nama variabel array
c. Subskrip / index array

Jenis Array (yang akan dipelajari) adalah :

a. Array Dimensi Satu
b. Array Dimensi Dua
c. Array Dimensi Tiga

Array dimensi satu adalah array yang terdiri dari n buah kolom atau array yang terdiri dari 1 subskrip array saja. Setiap elemen array satu dimensi dapat diakses melalui indeks yang terdapat di dalamnya.

Bentukpendeklarasian :

```
tipe_data  nama_array  [jumlah_elemen];
```

Contoh :

```
int ANGKA [10];
    atau
int ANGKA [10] = {2,3,4,5,6,1,7,3,55,3};
```

Array dua dimensi adalah array yang tersusun dalam bentuk baris dan kolom membentuk table, singkatnya dalam array dua dimensi kita membuat data yang tersusun dalam table.

Bentukpendeklarasian :

```
tipe_data  nama_array  [jumlah_elemen_baris] [jumlah_elemen_kolom];
```

Contoh :

```
Inty[2][2];
```

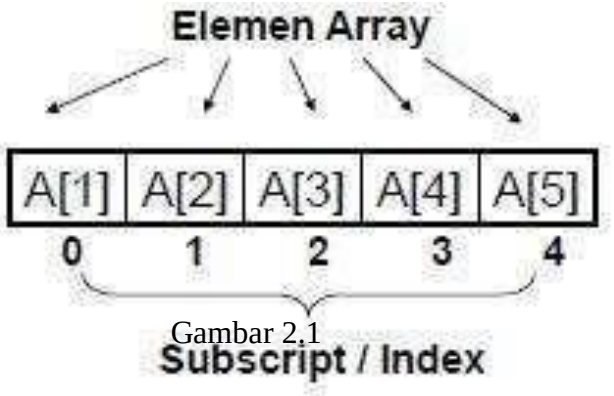
Array tiga dimensi adalah array yang memiliki banyak dimensi. Larik tersebut memiliki dimensi sesuai dengan kebutuhan

Bentukpendeklarasian:

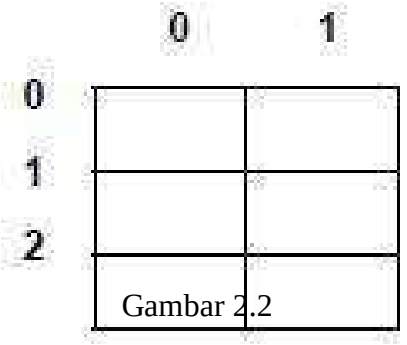
tipe_data nama_variabel[jumlah_baris][jumlah_kolom][panjang_karakter];

Contoh :

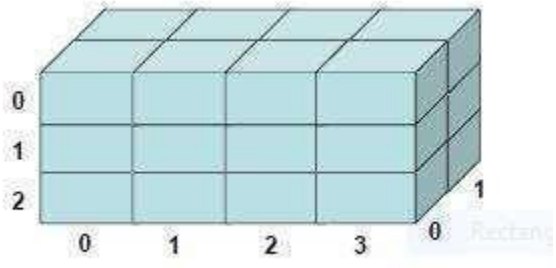
int x[2][2][2];



Gambar 2.1



Gambar 2.2



Gambar 2.3

Keterangan :

Gambar 2.1 = Array Dimensi 1
Gambar 2.2 = Array Dimensi 2
Gambar 2.3 = Array Dimensi 3

a. Program 1

Array Dimensi 1

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
    int a;
    int b[5];
    for(a=0;a<5;a++)
    {
        cout<<"Masukan Angka indeks ke ["<<a<<"] = ";
        cin>>b[a];
    }
    for(a=0;a<5;a++)
    {
    cout<<"\nTampilan angka ke "<<a+1<<" = ";
    cout<<b[a];
    }
    getch();
}
```

b. Program 2

Array Dimensi 2

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(){
    int matrik[3][4] = {{0,1,2,3},{4,5,6,7},{8,9,1,2}};
    int i,j;
    cout<<"Matrik 3x4\n";
    cout<<"==========\n";
    for(i=0;i<3;i++){
        for(j=0;j<4;j++){
          cout<<" "<<matrik[i][j];
        }
        cout<<endl;
    }
    getch();
}
```

c. Program 3
Array Dimensi 3

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
    int i,s;
    char h=64, nama[4][5][22]={
    "Rifky","Destian","Vienie","Andre","Mia",
    "Della","Vanya","Vio","Indri","Nilna",
    "Hakim","Bagus","Firman","Amin","Awan",
    "Ramdhan","Gilang","Lingga","Rizky","Kevin"
    };
    cout<<" Daftar Nama Kelompok : \n";
    for(i=0; i<4; i++)
    {
        ++h;
        cout<<" Kelompok "<<h;
        for(s=0; s<5; s++)
        {
            cout<<"\n\t"<<s+1<<". "<<nama[i][s];
        }
        cout<<"\n";
    }
    getch();
}
```

Buatlah sebuah program dari pernyataan berikut ini (pilih salah satu) :

1. Suatularikdengannamadaydideklarsikansebagaiberikut:
   inthari[] ={mon,tue,wed,thu,fri}

2. Suatu larik untuk menghitung rata-rata deret bilangan ganjil 1 sampai 100 yang habis dibagi 5.

PRAKTIKUM 3
SEARCHING

Searching merupakan proses fundamental dalam pengelolaan data.Proses pencarian adalah menemukan nilai (data) tertentu di dalam sekumpulan data yang bertipe sama (baik bertipe dasar atau bertipe bentukan).

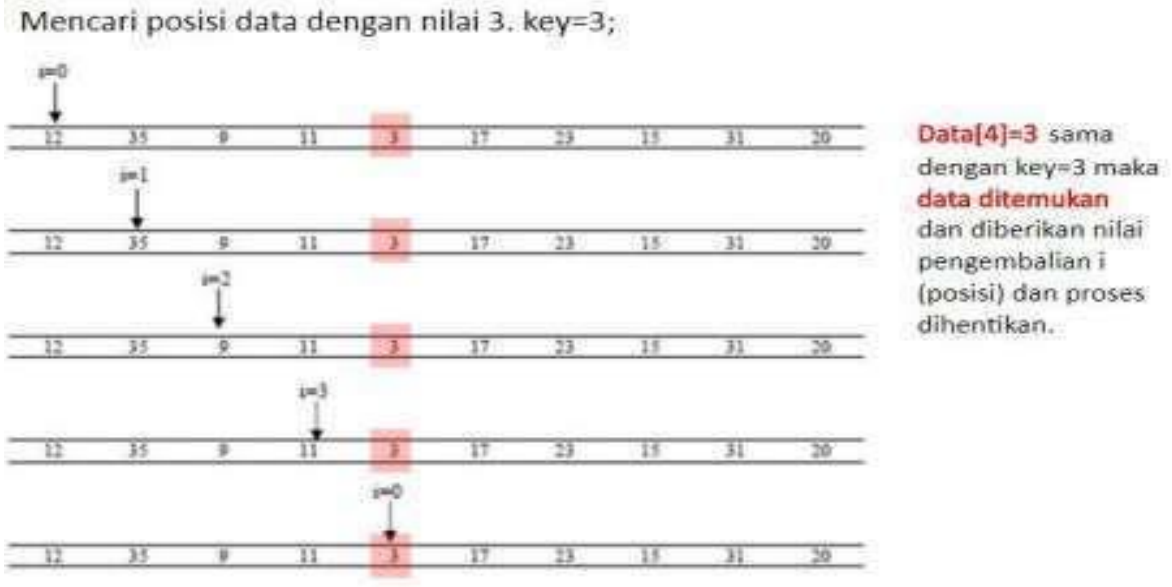
Metode Searching(yang akan dipelajari) adalah :

1. Sequential Search
2. Binary Search

Sequential Searchatau sering disebut pencarian linier.Menggunakan prinsip data yang ada di bandingkan satupersatu secara berurutan dengan yang dicari. Pada dasarnya, pencarian ini hanya melakukan pengulangan dari 1 sampai dengan jumlah data. Pada setiap perulangan , di bandingkan data ke-i dengan yang dicari. Apabilasama , berarti data telah ditemukan . Sebaliknya apabila sampai akhir pengulangan , tidak ada yang sama berarti data tidak ada.

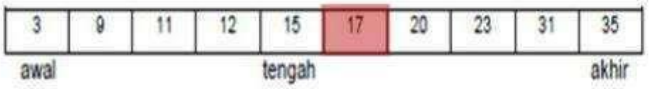
Binary search adalah sebuah algoritma pencarian dengan cara membagi data menjadi dua bagian setiap kali terjadi proses pencarian untuk menemukan nilai tertentu dalam sebuah larik (array) linear. Sebuah pencarian biner mencari nilai tengah (median), melakukan sebuah pembandingan untuk menentukan apakah nilai yang dicari ada sebelum atau sesudahnya, kemudian mencari setengah sisanya dengan cara yang sama.

1. **Sequential Search**

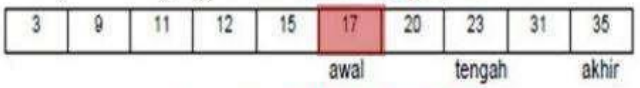


2. **Binary Search**

Contoh Data:
Misalnya data yang dicari **17**



Karena 17 > 15 (data tengah), maka: **awal = tengah + 1**



Karena 17 < 23 (data tengah), maka: **akhir = tengah – 1**

Karena 17 = 17 (data tengah), maka **KETEMU!**

a. Program 1

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;
int main(){
    system("cls");
    int data[8] = {8,27,9,20,10,1997,1996,100};
    int cari;
    int flag = 0;
    cout<<"Masukkan data yang ingin dicari = ";
    cin>>cari;
    for(int i=0;i<8;i++){
        if(data[i]==cari) flag=1;
    }
    if(flag==1)
    {
        cout<<"Data ada !\n";
    }
    else{
        cout<<"Data tidak ada !\n";
    }
}
```

b. Program 2

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
    int n,i,posisi,x,temu;
    int a[5];
    cout<<"Pencarian data dengan sequential search \n\n";
    cout<<"Banyak data : ";
    cin>>n;
    for(i=0;i<n;i++)
    {
        cout<<"Masukan angka : ";
        cin>>a[i];
    }
    cout<<"Data yang akan dicari : ";
    cin>>x;
    temu=0;
    i=0;
while((temu == 0)&&(i<n))
    {
    if(a[i]==x)
        {
            temu=1;
            posisi=i;
        }
    else
    i=i+1;
    }
    if(temu==1)
        cout<<"Ditemukan pada posisi "<<posisi+1<<" bernilai "<<x;
    else
        cout<<" Tidak ditemukan \n"<<x;
getch();
}
```

c. **Program 3**

```
#include <iostream>
#include <conio.h>
using namespace std;
int data[10]={1,3,4,7,12,25,40,65,78,90};
int binary_search(int cari)
{
    int a,b,c;
    int n=10;
    a=0;
    b=n-1;
    int temu=0;
    while (a<=b && temu==0)
    {
        c=(a+b)/2;
        if (data[c]==cari)
        temu=1;
        else
        if(cari<data[c])
        b=c-1;
        else a=c+1;
    }
    if(temu==1) return 1; else return 0;
}
int main()
{
    int cari, hasil;
    cout<<"Pencarian data dengan binary search \n\n";
    cout<<"Masukan data yang ingin di cari = ";
    cin>>cari;
    hasil = binary_search(cari);
    if(hasil==1)
    {
        cout<<"Data ada!"<<endl;
    }
    else
    if(hasil==0)
    cout<<"Data tidak ada!"<<endl;
getch();
}
```

Sel: 0 Lines: 39 Length: 664 Insert Done parsing in 0,016 seconds

Tuliskan dan jalankan dua program berikut ini dan jelaskan perbedaan dari kedua pemrogramantersebut!

a. Program 1

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;
int main()
{
    system("cls");
    int n,i,cari,arr[50];
    cout<<"Masukkan jumlah angka : ";
    cin>>n;
    cout<<"Masukkan "<<n<<" angka :\n";
    for(i=0;i<n;i++)
    {
        cin>>arr[i];
    }
    cout<<"Masukkan angka yang dicari : ";
    cin>>cari;
    for(i=0;i<n;i++)
    {
    if(arr[i]==cari)
        {
        cout<<cari<<" ditemukan pada lokasi "<<i+1;
        }
    }
    if(i==0)
    {
        cout<<"Tidak ditemukan!\n"<<cari<< "tidak termasuk dalam inputan";
    }
    getch();
}
```

b. Program 2

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;
int main()
{
    system("cls");
    int n, i, arr[50],cari,atas,bawah,tengah;
    cout<<"Masukkan jumlah angka : ";
    cin>>n;
    cout<<"Masukkan "<<n<<" angka :\n";
    for (i=0; i<n; i++)
    {
        cin>>arr[i];
    }
    cout<<"Masukkan angka yang akan dicari : ";
    cin>>cari;
    atas = 0;
    bawah = n-1;
    tengah = (atas+bawah)/2;
    while (atas <= bawah)
    {
        if(arr[tengah] < cari)
        {
            atas = tengah + 1;
        }
        else if(arr[tengah] == cari)
        {
            cout<<cari<<" ditemukan pada lokasi "<<tengah+1<<"\n";
            break;
        }
        else
        {
            bawah = tengah - 1;
        }
        tengah = (atas + atas)/2;
    }
    if(atas > bawah)
    {
        cout<<"Tidak ditemukan!\n"<<cari<<" tidak termasuk dalam inputan.";
    }
    getch();
}
```

PRAKTIKUM 4& 5
SORTING

Sorting adalah proses mengatur sekumpulan objek menurut aturan atau susunan tertentu. Urutan objek tersebut dapat menaik (ascending = dari data kecil ke data lebih besar) atau menurun (descending = dari data besar ke data lebih kecil).

Jenis Sorting(yang akan dipelajari) adalah :

1. Bubble sort (gelembung)
2. Selection sort (maksimum/minimun)
3. Insertion sort (sisip)
4. Quick sort
5. exchange sort

Bubble Sort merupakan metode sorting termudah, diberi nama "Bubble" karena proses pengurutan secara berangsur- angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda.

Jika pengurutan ascending :
jika elemen sekarang lebih besar dari elemen berikutnya maka kedua elemen tersebut ditukar

Jika pengurutan descending :
jika elemen sekarang lebih kecil dari elemen berikutnya, maka kedua elemen tersebut ditukar

Ketika satu proses telah selesai, maka bubble sort akan mengulangi proses, demikian seterusnya. Kapan berhentinya? Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan.Algoritma Bubble Sorting mudah dalam sintaks, tetapi lebih lambat dibandingkan dengan algoritma sorting yang lain

Selection Sort merupakan kombinasi antara sorting dan searching.Untuk setiap proses, akan dicari elemen-elemen yang belum diurutkan yang memiliki nilai terkecil atau terbesar akan dipertukarkan ke posisi yang tepat di dalam array.

Misalnya:
Untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan di indeks terkecil (data[0]), pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua (data[1])

Selama proses, pembandingan dan pengubahan hanya dilakukan pada indeks pembanding saja, pertukaran data secara fisik terjadi pada akhir proses.

Insertion Sort mirip dengan cara orang mengurutkan kartu, selembar demi selembar kartu diambil dan disisipkan (insert) ke tempat yang seharusnya.

- Pengurutan dimulai dari data ke-2 sampai dengan data terakhir, jika ditemukan data yang lebih kecil, maka akan ditempatkan (diinsert) diposisi yang seharusnya.
- Pada penyisipan elemen, maka elemen-elemen lain akan bergeser ke belakang

Quick Sort merupakan suatu algoritma pengurutan data yang menggunakan teknik pemecahan data menjadi partisi-partisi, sehingga metode ini disebut juga dengan nama partition exchange sort. Untuk memulai irterasi pengurutan, pertama-tama sebuah elemen dipilih dari data, kemudian elemen-elemen data akan diurutkan diatur sedemikian rupa, sehingga nilai variabel Sementara berada di suatu posisi ke I yang memenuhi kondisi sebagai berikut :

1. Semua elemen di posisi ke 1 sampai dengan ke I-1 adalah lebih kecil atau sama dengan Sementara.
2. Semua elemen di posisi ke I+1 sampai dengan ke N adalah lebih besar atau sama dengan Sementara.

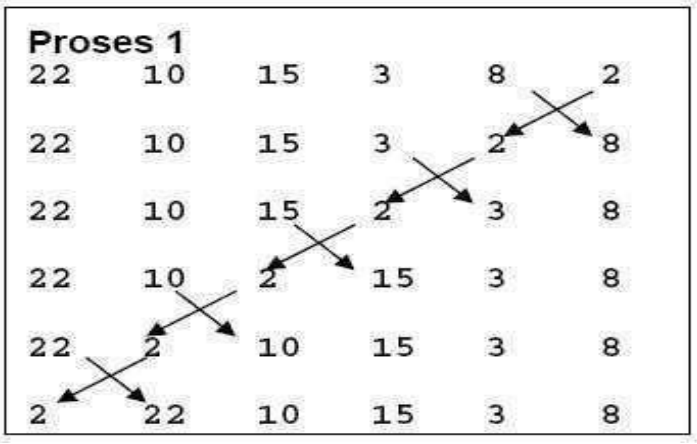
Exchange Sort sangat mirip dengan Bubble Sort, banyak yang mengatakan Bubble Sort sama dengan Exchange Sort.

Pebedaan :

dalam hal bagaimana membandingkan antar elemen-elemennya.

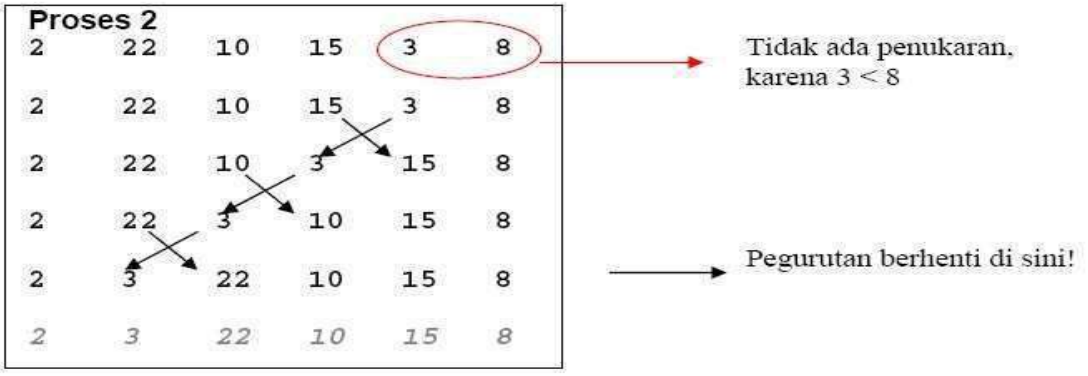
- Exchange sort membandingkan suatuelemen dengan elemen-elemen lainnya dalam array tersebut, dan melakukan pertukaran elemen jika perlu. Jadi ada elemen yang selalu menjadi elemen pusat (pivot).
- Sedangkan Bubble sort akan membandingkan elemen pertama/terakhir dengan elemen sebelumnya/sesudahnya, kemudian elemen tersebut itu akan menjadi pusat (pivot) untuk dibandingkan dengan elemen sebelumnya/sesudahnya lagi, begitu seterusnya.

1. **Bubble Sort**



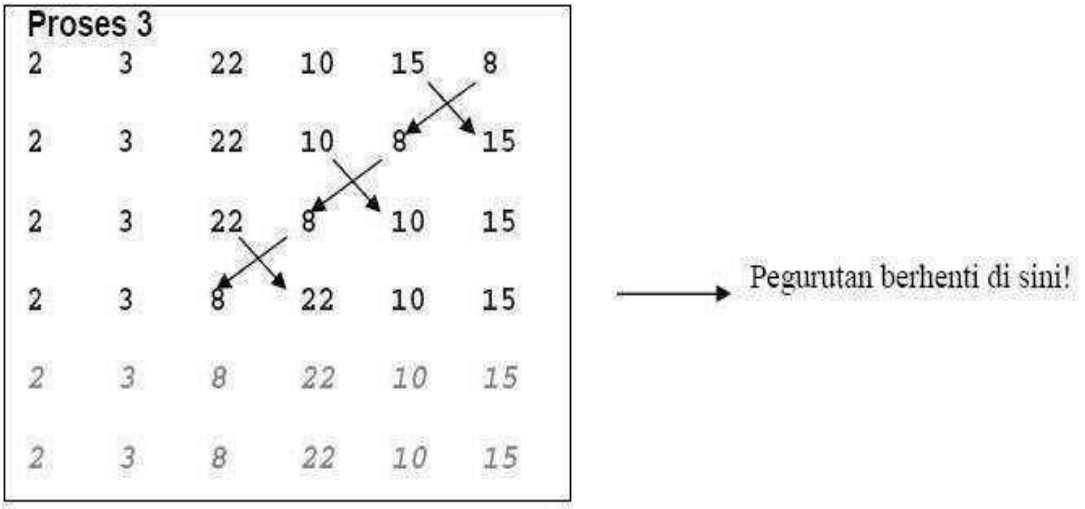
Gambar 1. Proses ke-1 algoritma Bubble Sorting

Pada gambar diatas, pegecekan dimulai dari data yang paling akhir, kemudiandibandingkan dengan data di depannya, jika data di depannya lebih besar maka akanditukar.

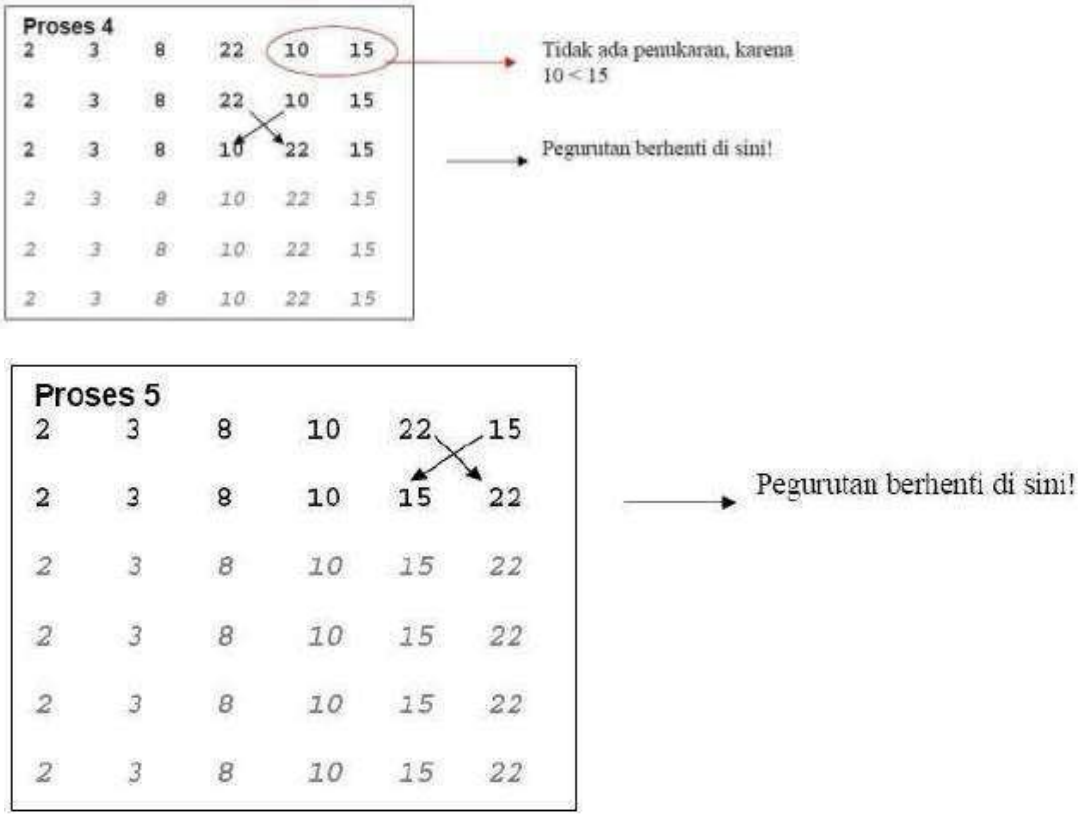


Gambar 2. Proses ke-2 algoritma Bubble Sorting

Pada proses kedua, pengecekan dilakukan sampai dengan data ke-2 karena datapertama pastisudah paling kecil.

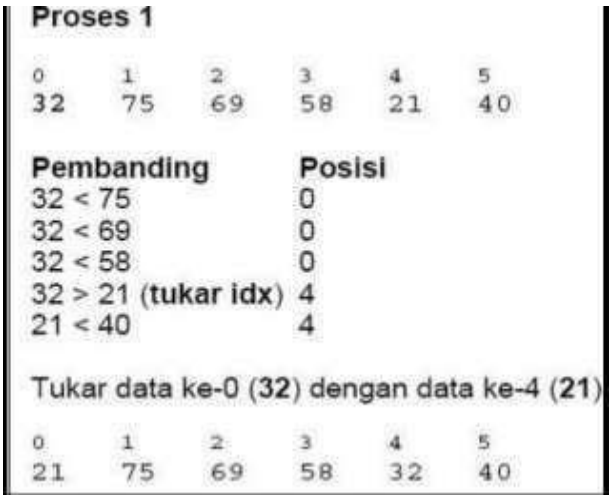


Gambar 3. Proses ke-3 algoritma Bubble Sorting

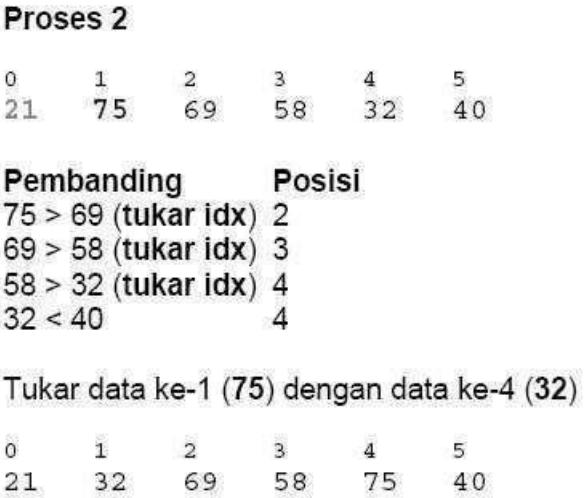


Gambar 5. Proses ke-5 algoritma Bubble Sorting

2. Selection Sort



Gambar 5. Proses ke-1 algoritma Selection Sort



Gambar 5. Proses ke-2 algoritma Selection Sort

Proses 3

0	1	2	3	4	5
21	32	69	58	75	40

Pembanding	Posisi
69 > 58 (**tukar idx**)	3
58 < 75	3
58 > 40	5

Tukar data ke-2 (**69**) dengan data ke-5 (**40**)

0	1	2	3	4	5
21	32	40	58	75	69

Gambar 3. Proses ke-3 algoritma Selection Sort

Proses 4

0	1	2	3	4	5
21	32	40	58	75	69

Pembanding	Posisi
58 < 75	3
58 < 69	3

Tukar data ke-3 (**58**) dengan data ke-3 (**58**)

0	1	2	3	4	5
21	32	40	58	75	69

Gambar 4. Proses ke-4 algoritma Selection Sort

Proses 5

0	1	2	3	4	5
21	32	40	58	75	69

Pembanding	Posisi
75 > 69	5

Tukar data ke-4 (**75**) dengan data ke-5 (**69**)

0	1	2	3	4	5
21	32	40	58	69	75

Gambar 5. Proses ke-5 algoritma Selection Sort

3. Insertion Sort

Proses 1

0	1	2	3	4	5
22	10	15	3	8	2

Temp	Cek	Geser
10	Temp<22?	Data ke-0 ke posisi 1

Temp menempati posisi ke -0

0	1	2	3	4	5
10	22	15	3	8	2

Proses 2

0	1	2	3	4	5
10	22	15	3	8	2

Temp	Cek	Geser
15	Temp<22	Data ke-1 ke posisi 2
15	Temp>10	-

Temp menempati posisi ke-1

0	1	2	3	4	5
10	15	22	3	8	2

Proses 3

0	1	2	3	4	5
10	15	22	3	8	2

Temp	Cek	Geser
3	Temp<22	Data ke-2 ke posisi 3
3	Temp<15	Data ke-1 ke posisi 2
3	Temp<10	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
3	10	15	22	8	2

Proses 4

0	1	2	3	4	5
3	10	15	22	8	2

Temp	Cek	Geser
8	Temp<22	Data ke-3 ke posisi 4
8	Temp<15	Data ke-2 ke posisi 3
8	Temp<10	Data ke-1 ke posisi 2
8	Temp>3	-

Temp menempati posisi ke-1

0	1	2	3	4	5
3	8	10	15	22	2

Proses 5

0	1	2	3	4	5
3	8	10	15	22	**2**

Temp	Cek	Geser
2	Temp<22	Data ke-4 ke posisi 5
2	Temp<15	Data ke-3 ke posisi 4
2	Temp<10	Data ke-2 ke posisi 3
2	Temp<8	Data ke-1 ke posisi 2
2	Temp<3	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
2	3	8	10	15	22

4. Quick sort

Contoh proses pengurutan Quick sort

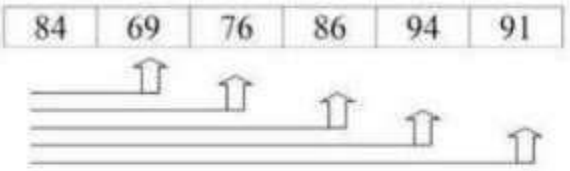
Data yang akan diurutkan secara **ascending** :

5 3 7 2

Data yang telah terurut secara **ascending** :

2 3 5 7

5. Exchange Sort



Proses 1

Pivot (Pusat)

84	69	76	86	94	91
84	69	76	86	94	91
84	69	76	**86**	94	91
86	69	76	**84**	**94**	91
94	69	76	84	**86**	91
94	69	76	84	86	91

Proses 2

Pivot (Pusat)

94	**69**	**76**	84	86	91
94	**76**	**69**	**84**	86	91
94	**84**	69	**76**	**86**	91
94	**86**	69	76	**84**	**91**
94	**91**	69	76	84	86

Proses 3

Pivot (Pusat)

94	91	**69**	**76**	84	86
94	91	**76**	**69**	**84**	86
94	91	**84**	69	**76**	**86**
94	91	**86**	69	76	84

Proses 4

Pivot (Pusat)

94	91	86	**69**	**76**	84
94	91	86	**76**	**69**	**84**
94	91	86	**84**	69	**76**

Proses 5

Pivot (Pusat)

94	91	86	84	**69**	**76**
94	91	86	84	**76**	**69**

a. Program

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
using namespace std;
int data[100],data2[100];
int n;

void tukar(int a,int b)
{
    int t;
    t = data[b];
    data[b] = data[a];
    data[a] = t;
}

void bubble_sort()
{
    for(int i=1;i<n;i++)
    {
        for(int j=n-1;j>=i;j--)
        {
            if(data[j]<data[j-1]) tukar(j,j-1);
        }
    }
    cout<<"bubble sort selesai!"<<endl;
}

void exchange_sort()
{
    for (int i=0; i<n-1; i++)
    {
        for(int j = (i+1); j<n; j++)
        {
            if (data [i] > data[j]) tukar(i,j);
        }
    }
    cout<<"exchange sort selesai!"<<endl;
}

void selection_sort()
{
    int pos,i,j;
        for(i=0;i<n-1;i++)
        {
        pos = i;
            for(j = i+1;j<n;j++)
            {
                if(data[j] < data[pos]) pos = j;
            }
        if(pos != i) tukar(pos,i);
        }
    cout<<"selection sort selesai!"<<endl;
}

void insertion_sort()
{
    int temp,i,j;
    for(i=1;i<n;i++)
    {
        temp = data[i];
        j = i -1;
        while(data[j]>temp && j>=0)
        {
            data[j+1] = data[j];
            j--;
        }
        data[j+1] = temp;
    }
    cout<<"insertion sort selesai!"<<endl;
```

```
70  }
71
72  void QuickSort(int L, int R)
73  {
74      int i, j;
75      int mid;
76      i = L;
77      j = R;
78      mid = data[(L+R) / 2];
79      do
80      {
81      while (data[i] < mid) i++;
82      while (data[j] > mid) j--;
83      if (i <= j)
84      {
85          tukar(i,j);
86          i++;
87          j--;
88      };
89      }
90      while (i < j);
91      if (L < j) QuickSort(L, j);
92      if (i < R) QuickSort(i, R);
93  }
94
95  void Input()
96  {
97      cout<<"Masukkan jumlah data = "; cin>>n;
98      for(int i=0;i<n;i++)
99      {
100         cout<<"Masukkan data ke-"<<(i+1)<<" = "; cin>>data[i];
101         data2[i] = data[i];
102     }
103 }
104
105 void Tampil()
106 {
107     cout<<"Data : "<<endl;
108     for(int i=0;i<n;i++)
109     {
110         cout<<data[i]<<" ";
111     }
112     cout<<endl;
113 }
114
115 void AcakLagi()
116 {
117     for(int i=0;i<n;i++)
118     {
119         data[i] = data2[i];
120     }
121     cout<<"Data sudah teracak!"<<endl;
122 }
123
124 int main()
125 {
126 int pil;
127 system("cls");
128     do
129     {
130     system("cls");
131     system("title Program Sorting");
132     cout<<" Menu Sorting"<<endl;
133     cout<<"**********************"<<endl;
134     cout<<" 1. Input Data"<<endl;
135     cout<<" 2. Bubble Sort"<<endl;
136     cout<<" 3. Exchange Sort"<<endl;
137     cout<<" 4. Selection Sort"<<endl;
138     cout<<" 5. Insertion Sort"<<endl;
```

```
139     cout<<" 6. Quick Sort"<<endl;
140     cout<<" 7. Tampilkan Data"<<endl;
141     cout<<" 8. Acak Data"<<endl;
142     cout<<" 9. Exit"<<endl;
143     cout<<"    Pilihan Anda = ";  cin>>pil;
144     switch(pil)
145     {
146         case 1:Input(); break;
147         case 2:bubble_sort(); break;
148         case 3:exchange_sort(); break;
149         case 4:selection_sort(); break;
150         case 5:insertion_sort(); break;
151         case 6:QuickSort(0,n-1);
152         cout<<"quick sort selesai!"<<endl;
153         break;
154         case 7:Tampil(); break;
155         case 8:AcakLagi(); break;
156     }
157     getch();
158     }
159     while(pil!=9);
160 }
```

1. Jelaskan maksud sebuah program dalam pelaksanaan praktikum diatas !
2. Buatlah sebuah program yang menunjukkan aktivitas sorting dengan menggunakan salah satu metode sorting dan jelaskan kenapa menggunakan metode tersebut !

PRAKTIKUM 6& 7
LINKED LIST

Linked List adalah suatu cara untuk menyimpan data dengan struktur sehingga programmer dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan data kapan saja diperlukan. Linked list dikenal juga dengan sebutan senarai berantai adalah stuktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada linked list disebut Node. Biasanya dalam suatu linked list, terdapat istilah head dan tail .

1. Head adalah elemen yang berada pada posisi pertama dalam suatu linked list
2. Tail adalah elemen yang berada pada posisi terakhir dalam suatu linked list.

Jenis Linked List (yang akan dipelajari) adalah :

1. Single Linked List
2. Double Linked List
3. Circular Linked List
4. Multiple Linked List

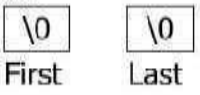
Ada 5 proses dasar dalam Linked List :

1. ProsesInisialisasi

- Proses awal → menyatakan Linked List belum ada.
- Algoritma :

```
First = Null;
Last = Null;
```

- Ilustrasi Proses :



2. Membuat simpul baru.

- Instruksi :

```
P = (simpul *) malloc(sizeof(simpul));
```

- Algoritma :

```
void Buat_Simpul(int x)
{
        P = (simpul *) malloc(sizeof(simpul));
        if (P != NULL)
        {       P -> Info = x; }
        else
                cout<<"Simpul gagal dibuat ";
}
```

3. Membuat simpul awal.

• Algoritma :

```
void Awal()
{
        First = P;
        Last = P;
        P -> Link = NULL;
}
```

Syarat :

1. Linked List belum ada.
2. Sudah ada simpul yang akan dijadikan simpul awal.

4. Menambahkan simpul baru ke dalam Linked List (INSERT)

Syarat :

1.Linked List sudah ada.

2.Sudah ada simpul yang akan ditambahkan ke Linked List.

a. Insert Kanan/Akhir

Algoritma :

```
void Ins_Akhir()
{
        Last -> Link = P;
        Last = P;
        P -> Link = NULL;
}
```

b. Insert Kiri/Awal

Algoritma :

```
void Ins_Awal()
{
        P -> Link = First;
        First = P;
}
```

c. Insert Tengah

Algoritma :

```
void Ins_Tengah()
{
        P -> Link = Q ->Link;
        Q -> Link = P;
}
```

5. Menghapus sebuah simpul dari Linked List (DELETE)

Syarat :

1. Linked List sudah ada.

a. Delete Kanan/Akhir

Algoritma :

```
void Del_Akhir()
{
        free(Last);
        Last = Q;
        Last -> Link = NULL;
}
```

b. Delete Kiri/Awal

Algoritma :

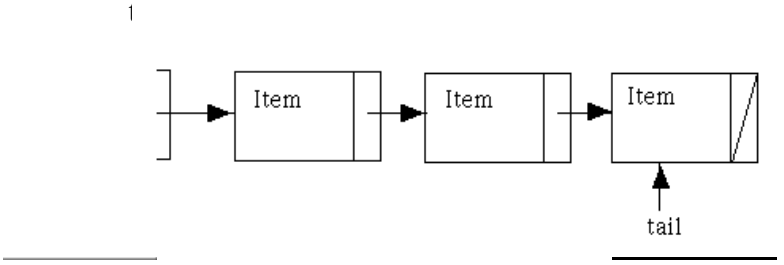
```
void Del_Awal()
{
        Q = First;
        First = Q -> Link;
        free(Q);
}
```

c. Delete Tengah

Algoritma :

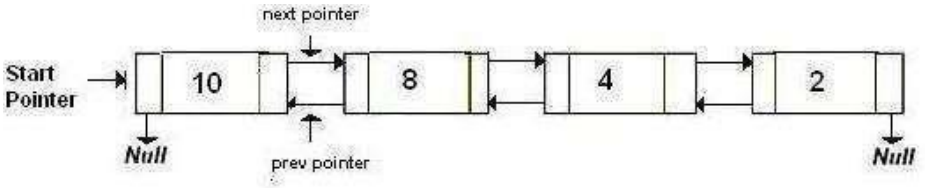
```
void Del_Tengah()
{
        R = Q->Link ;
        Q->Link = R->Link;
        free(R);
}
```

Single Linked List Dimana pointer l menunjuk ke NULL.



Gambar . Ilustrasi Single Linked List

Double Linked List merupakan suatu linked list yang memiliki dua variabel pointer yaitu pointer yang menunjuk ke node selanjutnya dan pointer yang menunjuk ke node sebelumnya. Setiap head dan tailnya juga menunjuk ke NULL.

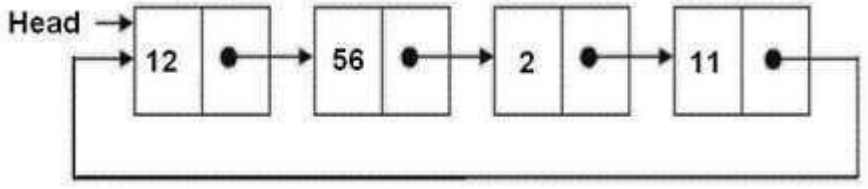


Gambar . Ilustrasi Double Linked List

Circular Linked List merupakan suatu linked list dimana tail (node terakhir) menunjuk ke head (node pertama). Jadi tidak ada pointer yang menunjuk NULL.

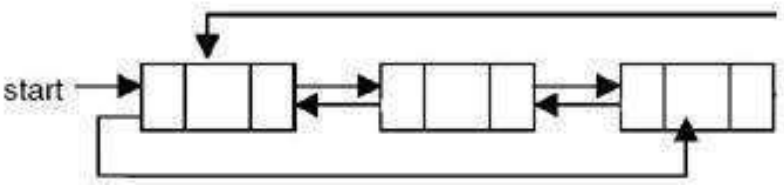
Ada 2 jenis CircularLinked List, yaitu :

1. Circular Single Linked List



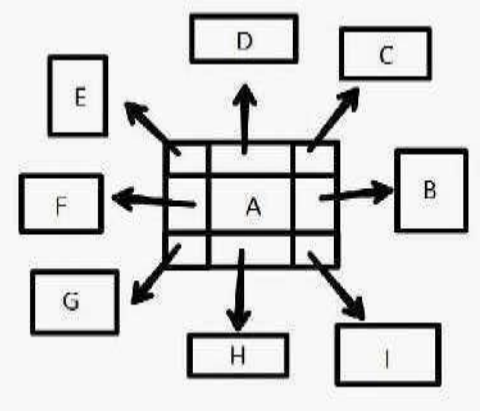
Gambar . Ilustrasi Circular Single Linked List

2. Circular Double Linked List



Gambar . Ilustrasi Circular Double Linked List

Multiple Linked List merupakan suatu linked list yang memiliki lebih dar 2 buat variabel pointer.



Gambar . Ilustrasi Multiple Linked List

a. Program 1

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
using namespace std;
struct TNode
{
    int data;
    TNode *next;
};
TNode *head, *tail;
void init()
{
    head = NULL;
    tail = NULL;
}
int isEmpty()
{
    if(tail == NULL) return 1;
    else return 0;
}
void insertDepan(int databaru)
{
    TNode *baru;
    baru = new TNode;
    baru->data = databaru;
    baru->next = NULL;

    if(isEmpty()==1){
        head=tail=baru;
        tail->next=NULL;
    }
    else {
        baru->next = head;
        head = baru;
    }
      cout<<"Data masuk\n";
      system("CLS");
}

void insertBelakang(int databaru)
{
    TNode *baru,*bantu;
    baru = new TNode;
    baru->data = databaru;
    baru->next = NULL;

    if(isEmpty()==1){
        head=baru;
        tail=baru;
        tail->next = NULL;
```

```
51      }
52      else{
53          tail->next = baru;
54          tail=baru;
55      }
56      cout<<"Data masuk\n";
57      system("CLS");
58  }
59
60  void tampil()
61  {
62      TNode *bantu;
63      bantu = head;
64      if(isEmpty()==0){
65          cout<<"Data yang masuk: ";
66          while(bantu!=NULL){
67              cout<<bantu->data<<" ";
68              bantu=bantu->next;
69          }
70      }
71      else cout<<"Data yang masuk: Masih kosong\n";
72    }
73
74  void hapusDepan(){
76      int d;
77      if (isEmpty()==0){
78          if(head!=tail){
79              hapus = head;
80              d = hapus->data;
81              head = head->next;
82              delete hapus;
83          }
84          else {
85          d = tail->data;
86          head=tail=NULL;
87          }
88      cout<<d<<"terhapus";
89      }
90      else cout<<"Masih kosong\n";
91      system("CLS");
92  }
93
94  void hapusBelakang()
95  {
96      TNode *bantu,*hapus;
97      int d;
98      if (isEmpty()==0){
99      bantu = head;
100         if(head!=tail){
101             while(bantu->next!=tail){
102             bantu = bantu->next;
103             }
104         hapus = tail;
105         tail=bantu;
106         d = hapus->data;
107         delete hapus;
108         tail->next = NULL;
109         }
110         else{
111         d = tail->data;
112         head=tail=NULL;
113         }
114         cout<<d<<" terhapus\n";
115     }
116     else cout<<"Masih kosong\n";
117   system("CLS");
118 }
119
120 main()
121 {
122     int pil,databaru;
123     cout<<"*---------------------*"<<endl;
124     cout<<"* Single Linked List  "<<endl;
125     cout<<"*---------------------*"<<endl;
```

```
126     do{
127         cout<<"\n";
128         cout<<"\n**********************";
129         cout<<"\n1. Insert Depan";
130         cout<<"\n2. Insert Belakang";
131         cout<<"\n3. Delete Depan";
132         cout<<"\n4. Delete Belakang";
133         cout<<"\n5. Tampil Data";
134         cout<<"\n\nSilahkan Masukan Pilihan Anda :";cin>>pil;
135         cout<<"\n";
136         switch (pil){
137             case 1:
138             {
139                 cout<<"Masukkan Data = ";
140                 cin>>databaru;
141                 insertDepan(databaru);
142                 break;
143             }
144             case 2:
145             {
146                 cout<<"Masukkan Data = ";
147                 cin>>databaru;
148                 insertBelakang(databaru);
149                 break;
151             case 3:
152             {
153                 hapusDepan();
154                 break;
155             }
156             case 4:
157             {
158                 hapusBelakang();
159                 break;
160             }
161             case 5:
162             {
163                 tampil();
164                 break;
165             }
166             default :
167             {
168                 cout<<"\n Maaf, Tidak ada dalam pilihan";
169             }
170         }
171     }
172     while(pil>=1 && pil<= 5);
173 }
```

b. Program 2

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<conio.h>
using namespace std;
typedef struct simpul node;
struct simpul
{
    int data;
    node *prev;
    node *next;
};

node *head = NULL, *tail = NULL, *baru;

void allocate_node (int x)
{
    baru = (node *) malloc (sizeof(node));
    if(baru==NULL)
    {
        printf("Alokasi Gagal\n");
        exit(1);
    }
    else
    {
```

```
        baru->data=x;
        baru->next=NULL;
        baru->prev=NULL;
    }
}

//head = tail = baru;


void buat_list()
{
    printf("Data Masih Kosong, List akan dibuat dengan data tersebut\n");
    system("PAUSE");
    head = tail = baru;
}

void sisip_awal()
{
    if(head==NULL && tail==NULL)
        buat_list();
    else
    {
        baru->next=head;
        head->prev=baru;
        head=baru;
    }
}

void sisip_akhir()
{
    if(head==NULL && tail==NULL)
        buat_list();
    else
    {
        baru->prev=tail;
        tail->next=baru;
        tail=baru;
    }

}

void sisip_sebelum(int s)
{
    node *before=head;
    int cek=0;
    if(before->next==NULL)
        cek=1;
    else
    {
        while(before->next->data!=s)
        {
            before=before->next;
            if(before->next==NULL)
            {
                cek = 1;
                break;
            }
        }
    }
    if(cek==0)
    {
        baru->prev = before;
        baru->next = before->next;
        before->next->prev = baru;
        before->next = baru;
    }
    else
        void gatot();
}

void sisip_sesudah(int s)
{
    node *after=tail;
    int cek=0;
    if(after->prev == NULL)
```

```
101         cek=1;
102     else
103     {
104         while(after->prev->data!=x)
105         {
106             after=after->prev;
107             if(after->prev==NULL)
108             {
109                 cek = 1;
110                 break;
111             }
112         }
113     }
114     if(cek==0)
115     {
116         baru->next = after;
117         baru->prev = after->next;
118         after->prev->next = baru;
119         after->prev = baru;
120     }
121     else
122         void gatot();
123 }
124
125 void free node(node *p)
126 {
127     free(p);
128     printf("DATA TERHAPUS\n");
129 }
130
131 void hapus_satu()
132 {
133     node *hapus = head;
134     head = NULL;
135     tail = NULL;
136     free_node(hapus);
137 }
138
139 void hapus awal()
140 {
141     node *hapus = head;
142     head->next->prev = NULL;
143     head =  head->next;
144     free_node(hapus);
145 }
146
147 void hapus_akhir()
148 {
149     node *hapus = tail;
150     tail->prev->next = NULL;
151     tail =  tail->prev;
152     free node(hapus);
153 }
154
155 void hapus_tengah(int x)
156 {
157     node *hapus=head;
158     int cek=0;
159     while(hapus->data!=x)
160     {
161         if(hapus->next==NULL)
162         {
163             cek=1;
164             break;
165         }
166         hapus = hapus->next;
167     }
168     if(cek==0)
169     {
170         hapus->prev->next = hapus->next;
171         hapus->next->prev = hapus->prev;
172         free node(hapus);
173     }
174     else
175         void gadel();
```

```
176 }
177
178 void gatot()
179 {
180     printf("SIMPUL BARU TIDAK DAPAT DISISIPKAN\n");
181     //system("PAUSE");
182 }
183
184 void gadel()
185 {
186     printf("TIDAK ADA DATA YANG DIHAPUS\n");
187     //system("PAUSE");
188 }
189
190 void tampil()
191 {
192     node *p= head;
193     printf("\nData Simpul ==>  ");
194     while(p!=NULL)
195     {
196         printf("%d ", p->data);
197         p=p->next;
198     }
199     printf("\n\n");
200 }
201
202 int main()
203 {
204     head=baru;
205     int pilih, data, s;
206     char lagi='y';
207     while(lagi=='y')
208     {
209         system("CLS");
210         void awal();
211         printf("Program Double Linked List\n");
212         printf("**************************");
213         tampil();
214         printf("Menu Pilihan : \n");
215         printf("1. Sisip Awal\n");
216         printf("2. Sisip Akhir\n");
217         printf("3. Sisip Sebelum Simpul\n");
218         printf("4. Sisip Sesudah Simpul\n");
219         printf("5. Hapus Awal\n");
220         printf("6. Hapus Akhir\n");
221         printf("7. Hapus Tengah\n");
222         printf("\nPilih No           : ");
223         scanf("%d", &pilih);
224         switch(pilih)
225         {
226         case 1 :
227             printf("Masukkan data       : ");
228             scanf("%d", &data);
229             allocate_node(data);
230             sisip_awal();
231             break;
232         case 2 :
233             printf("Masukkan data       : ");
234             scanf("%d", &data);
235             allocate_node(data);
236             sisip_akhir();
237             break;
238         case 3 :
239             printf("Masukkan data       : ");
240             scanf("%d", &data);
241             allocate_node(data);
242             if(head==NULL && tail==NULL)
243                 buat_list();
244             else
245             {
246                 printf("Dimasukkan sebelum  : ");
247                 scanf("%d",&s);
248                 if(s==head->data)
249                     sisip_awal();
250                 else
```

```
            sisip_sebelum(s);
        }
        break;
    case 4 :
        printf("Masukkan data       : ");
        scanf("%d", &data);
        allocate_node(data);
        if(head==NULL && tail==NULL)
            buat_list();
        else
        {
            printf("Dimasukkan sesudah  : ");
            scanf("%d",&s);
            if(s==tail->data)
                sisip_akhir();
            else
                sisip_sesudah(s);
        }
        break;
    case 5 :
        if(head == NULL && tail == NULL)
            gadel();
        else if(head == tail)
            hapus_satu();
        else
            hapus_awal();
        break;
    case 6 :
        if(head == NULL && tail == NULL)
            gadel();
        else if(head == tail)
            hapus_satu();
        else
            hapus_akhir();
        break;
    case 7 :
        printf("Data yang dihapus   : ");
        scanf("%d", &data);
        if(head == NULL && tail == NULL)
            gadel();
        else if(data == head->data && data == tail->data && head == tail)
            hapus_satu();
        else if(data == head->data)
            hapus_awal();
        else if(data == tail->data)
            hapus_akhir();
        else
            hapus_tengah(data);
        break;
    }
    fflush(stdin);
    printf("Lagi (y/t) ? ");
    scanf("%c", &lagi);
  }
}
```

1. Jelaskan maksud program dalam pelaksanaan praktikum diatas (pilih salah satu) !
2. Buatlah sebuah program linked list (metode bebas) yang berisikan menu sebagai berikut :
   a. Insert depan
   b. Insert tengah
   c. Insert belakang
   d. Delete depan
   e. Delete tengah
   f. Delete belakang
   g. Tampilkan data

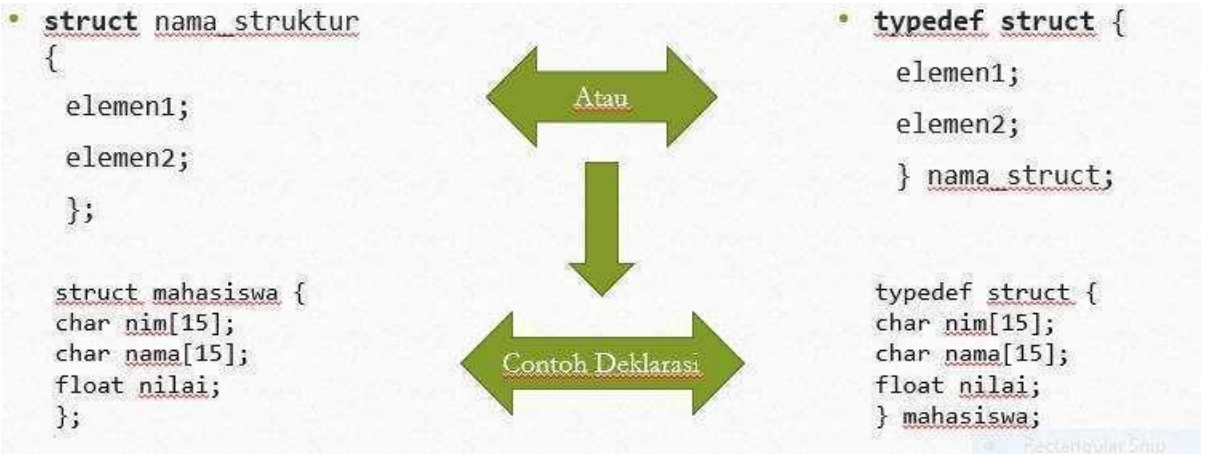
h. Keluar

PRAKTIKUM 8
STRUCT

Struktur adalah tipe data yang dapat menyimpan sejumlah data yang memiliki tipe data berbeda. Variabel yang menyusun Struktur dinamakan elemen struktur.

Hal yang perlu diingat :

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



Pengaksesanelemenstructdilakukansecaraindividualdenganmenyebutkannamavariablestrukturdiikutidengantandatitik “.”.

Perhatikan contoh berikut:

```
mahasiswa mhs;
cout<<"Masukkan NIM : ";
cin >> mhs.nim;
cout<<"Masukkan Nama : ";
cin >>mhs.nama;
cout<<"Masukkan Nilai Akhir : "<< mhs.nilai;
```

PRAKTIKUM 8

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;

struct mahasiswa
{
    char nim[15];
    char nama[15];
    float nilai;
};
int main()
{
    system("cls");
    mahasiswa mhs;
    cout<<"masukan NIM = ";
    cin>>mhs.nim;
    cout<<"masukan Nama = ";
    cin>>mhs.nama;
    cout<<"masukan Nilai Akhir = ";
    cin>>mhs.nilai;
    cout<<endl<<endl;
    cout<<"NIM = "<<mhs.nim<<endl;
    cout<<"Nama = "<<mhs.nama<<endl;
    cout<<"Nilai Akhir = "<<mhs.nilai<<endl;
    getch();
}
```

1. Buatlah sebuah program untuk menghitung gaji harian,
   gaji perjam = 50000,
   bila jumlah jam kerja lebih dari 7 jam, maka lebihnya dianggap lembur dan gaji perjam lembur = 15 x gaji perjam.
   Gunakan Struct untuk membuat program tersebut !

PRAKTIKUM 9
POINTER

Pointer adalah suatu variabel penunjuk yang menunjuk pada suatu alamat memori komputer tertentu. Variabel pointer yang tidak menunjuk pada nilai apapun berartimemiliki nilai NULL, dan disebut sebagai **dangling pointer** karena nilainya tidakdiinisialisasi dan tidak dapat diprediksi.

1. Operator Alamat / Dereference Operator(&)

Setiap variabel yang dideklarasikan, disimpan dalam sebuah lokasi memori dan pengguna biasanya tidak mengetahui di alamat mana data tersebut disimpan. Dalam C++, untuk mengetahui alamat tempat penyimpanan data, dapat digunakan tanda ampersand(&) yang dapat diartikan "alamat".

Contoh :

```
Bil1 = &Bil2;
```

dibaca: isi variabel bil1 sama dengan alamat bil2

2. Operator Reference (*)

Penggunaan operator ini, berarti mengakses nilai sebuah alamat yang ditunjukolehvariabel pointer.

Contoh :

```
Bil1=*Bil2;
```

dibaca: bil1 sama dengan nilai yang ditunjuk oleh bil2.

1. Operasi penugasan

Nilai dari suatu variabel pointer dapat disalin ke variabel pointer yang lain.

2. Operasi aritmatika

- Suatu variabel pointer hanya dapat dilakukan operasi aritmatika dengannilai integer saja.
- Operasi yang biasa dilakukan adalah operasi penambahan danpengurangan.
- Operasi penambahan dengan suatu nilai menunjukkan lokasi databerikutnya (index selanjutnya) dalam memori.
- Begitu juga operasi pengurangan.

3. Operasi Logika

PRAKTIKUM 9

```
#include <iostream>
#include <conio.h>
using namespace std;
int main() {
    int x, y; int *px;
    x = 89;
    y = x;
    px = &x;
    cout << "Nilai x = " << x << endl;
    cout << "Nilai y = " << y << endl;
    cout << "Alamat px = " << px << endl;
    cout << "Nilai px = " << *px << endl;
    getch();
}
```

b. Program 2

```
#include <iostream>
#include <conio.h>
using namespace std;
int main() {
int x;
int *px; //pointer ke variabel
int **ppx; //pointer ke pointer

x = 175;
px = &x;
ppx = &px;
cout << "Nilai x = " << x << endl;
cout << "Nilai px = " << *px << endl;
cout << "Nilai ppx = " << **ppx << endl;

getch();
}
```

c. Program 3

```
#include <iostream>
#include <conio.h>
#define MAX 5
using namespace std;
int main() {
    int a[MAX];
    int *pa; pa = a; //atau pa = &a[0]

    for (int i = 0; i < MAX; i++) {
    cout << "Masukkan Nilai " << i+1 << " : ";
    cin >> a[i];
    }
    cout << endl;
    for (int i = 0; i < MAX; i++) {
    cout << "Nilai a[" << i << "] = " << *pa << endl;
    pa++;
    }
getch();
}
```

d. Program 4

```
1  #include <iostream>
2  #include <conio.h>
3  #define MAX 5
4  using namespace std;
5  int main() {
6  char nama[] = "Mia Kastina";
7  char *pNama = nama;
8
9  cout << "Nama = " << nama << endl;
10 cout << "pNama = " << pNama << endl;
11 pNama += 4; cout << "\nSetelah pNama += 4" << endl;
12 cout << "Nama = " << nama << endl;
13 cout << "pNama = " << pNama << endl;
14
15 getch();
16 }
```

1. Deskripsikan program pelaksanaan praktikum tersebut !
2. Buatlah sebuah programyang menampilkan kalimat terbalik dari suatu kalimat yang telah diinputkan maupun data yang telah ada dalam program. Menggunakan pointer dan beberapa fungsi.

PRAKTIKUM 10& 11
STACK

Stack (Tumpukan) adalah list linier yang dikenali elemen puncaknya (TOP) dan Aturan penyisipan dan penghapusan elemennya tertentu. Penyisipan selalu dilakukan "di atas" TOP dan Penghapusan selalu dilakukan pada TOP.

Hal yang perlu diingat :

1. Stack disebut juga tumpukan dimana data hanya dapat dimasukkan dan diambil dari satu sisi.
2. Stack bersifat LIFO (Last In First Out).
   Benda yang terakhir masuk ke dalam stack akan menjadi benda pertama yang dikeluarkan dari stack.

a. 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
```

b. Mendefinisikan max_stack untuk maksimum isi stack
#define max_stack 15

c. Membuat variable array sebagai implementasi stack
stack tumpuk;

1. Menambah (push)

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

2. Mengambil (pop)

```
void pop()
{
     printf ("data %sterambil",tumpuk.data[tumpuk.top]);
```

3. **Megecek apakah stack penuh (isFull)**

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

4. **Mengecek apakah stack kosong (isEmpty)**

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

5. **Membersihkan stack (clear)**

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

6. **Mencetak isi stack (print)**

```
void print()
{
      for (int i=tumpuk.top;i>=0;i--)
      {
      Cout<<tumpuk.data[i];
      }
}
```

Program

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
#define max 10
using namespace std;

struct Tumpukan
{
    int atas;
    int data[max];
}

T;

void awal()
{
    T.atas=-1;
}

int kosong()
{
    if(T.atas==-1)
    return 1;
    else
    return 0;
}

int penuh()
{
    if(T.atas==max-1)
    return 1;
    else
    return 0;
}

void input(int data)
{
    if(kosong()==1)
    {
        T.atas++;
        T.data[T.atas]=data;
        cout<<"Data "<<T.data[T.atas]<<" masuk ke stack";
    }

    else if(penuh()==0)
    {
        T.atas++;
        T.data[T.atas]=data;
        cout<<"Data "<<T.data[T.atas]<<" masuk ke stack";
    }

    else
        cout<<"Tumpukan penuh";
}

void hapus()
{
   if(kosong()==0)
   {
        cout<<"Data teratas sudah terambil";
        T.atas--;
   }

   else
        cout<<"Data kosong";
}

void tampil()
{
    if(kosong()==0)
    {
        for(int i=T.atas;i>=0;i--)
```

1.

```
        {
            cout<<"\nTumpukan ke "<<i<<"="<<T.data[i];
        }
    }

    else
        cout<<"Tumpukan kosong";
}

void bersih()
{
    T.atas=-1;

    cout<<"Tumpukan kosong!";
}

int main()
{
    int pil,data;
    awal();
    do
    {
        system("cls");
        cout<<"1. Input\n2. Hapus\n3. Tampil\n4. Bersihkan\n5. Keluar\nMasukkan pilihan :";
        cin>>pil;
        switch(pil)
        {
            case 1:cout<<"Masukkan data = ";
            cin>>data;
            input(data);
            break;

            case 2:hapus();
            break;

            case 3:tampil();
            break;

            case 4:bersih();
            break;

            case 5: cout<<"Terimakasih, tekan enter untuk keluar";
        }

        getch();
    }
        while(pil!=5);
}
```

Deskripsikan program pelaksanaan praktikum tersebut !

2. Buatlah sebuah program yang menunjukkan program tersebut sebuah program stack. Gunakan tampilan menarik dan penjelasan apa saja yang digunakan dalam program tersebut !

PRAKTIKUM 12& 13
QUEUE

Queue (Antrian) adalah suatu bentuk khusus dari List Linier dengan operasi penyisipan (*insertion*) hanya diperbolehkan pada salah satu sisi, yang disebut sisi belakang (REAR), dan operasi penghapusan (*deletion*) hanya diperbolehkan pada sisi yang lainnya

Hal yang perlu diingat :

1. Queue disebut juga antrian dimana data masuk di satu sisi dan keluar di sisi yang lain.
2. Queue bersifat FIFO (First In First Out).

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

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

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

3. Membuat variable array sebagai implementasi queue
queue antri;

1. 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");
     }
```

2. 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--;
}
```

3. isEmpty (mengecek apakah antrian kosong)

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

4. isFull (mengecek apakah antrian penuh)

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

5. Clear (membersihkan seluruh isi antrian)

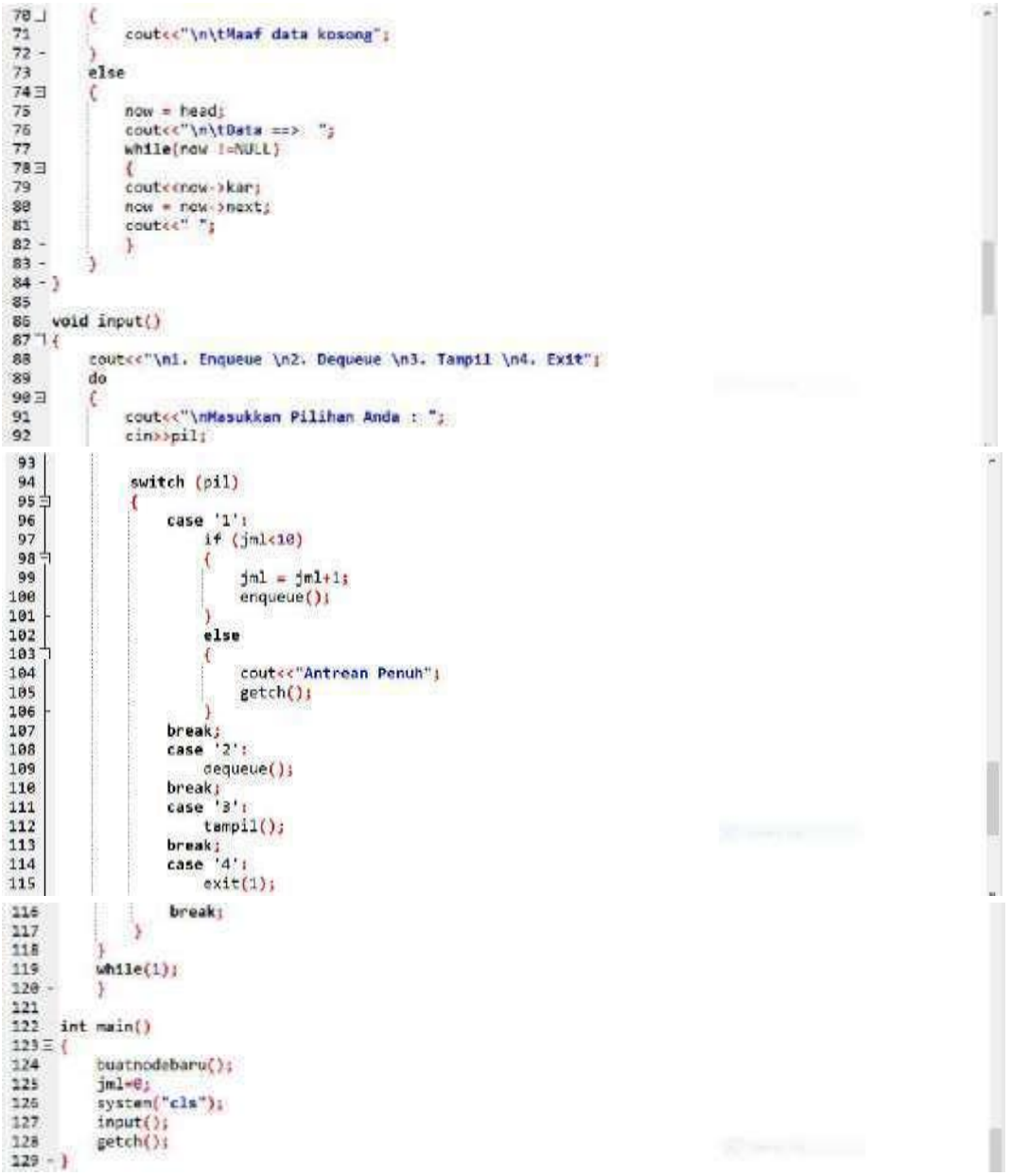
```
void clear()
{
     antri.head=antri.tail=-1;
     printf("semua data
terhapus.");
}
```

6. Print(mencetak seluruh isi antrian)

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

Program

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
using namespace std;
char pil;
int jml;

struct node
{
    char kar;
    node *next;
};

node *tail, *now ,*head;

void buatnodebaru()
{
    tail=NULL;
    head=NULL;
}

void push(char ch)
{
    now = new node;
    if(head == NULL)
    {
        now->kar=ch;
        now->next=NULL;
        tail = now;
        head = now;
    }
    else
    {
        now->kar=ch;
        now->next=NULL;
        tail->next=now;
        tail=now;
    }
}

void enqueue()
{
    int i;
    char temp;
    cout<<"Masukkan 6 karakter : ";
    cin>>temp;
    push(temp);
}

void dequeue()
{
    if(head==NULL)
    {
    cout<<"Antrean Kosong..";
    }
    else
    {
        jml = jml - 1;
        char delete_element = head->kar;
        node *temp;
        temp = head;
        head = head->next;
        delete temp;
    }
}

void tampil()
{
    if (head == NULL)
```



```
    {
        cout<<"\n\tMaaf data kosong";
    }
    else
    {
        now = head;
        cout<<"\n\tData ==>  ";
        while(now !=NULL)
        {
        cout<<now->kar;
        now = now->next;
        cout<<" ";
        }
    }
}

void input()
{
    cout<<"\n1. Enqueue \n2. Dequeue \n3. Tampil \n4. Exit";
    do
    {
        cout<<"\nMasukkan Pilihan Anda : ";
        cin>>pil;

        switch (pil)
        {
            case '1':
                if (jml<10)
                {
                    jml = jml+1;
                    enqueue();
                }
                else
                {
                    cout<<"Antrean Penuh";
                    getch();
                }
            break;
            case '2':
                dequeue();
            break;
            case '3':
                tampil();
            break;
            case '4':
                exit(1);
            break;
        }
    }
    while(1);
    }

int main()
{
    buatnodebaru();
    jml=0;
    system("cls");
    input();
    getch();
}
```

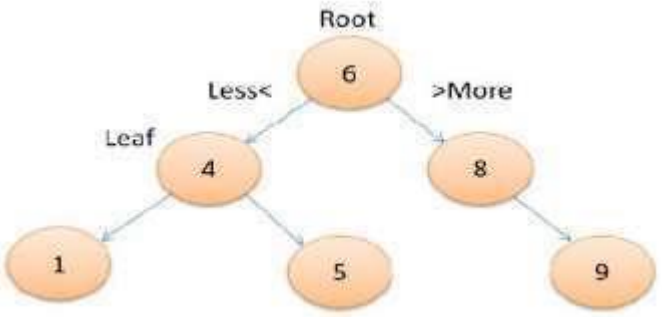
1. Deskripsikan program pelaksanaan praktikum tersebut !
2. Buatlah sebuah program yang menunjukkan program tersebut sebuah program queue. Gunakan tampilan menarik dan penjelasan apa saja yang digunakan dalam program tersebut !

PRAKTIKUM 14
BINARY TREE

Binary Tree (Pohon Biner) yaitu pohon yang setiap simpul/node-nya paling banyak mempunyai dua buah subpohon.

Contoh implementasi :
untuk membuat pohon silsilah keluarga, ungkapan aritmatika yang setiap operatornya dipasang sebagai simpul pencabangan dan operand-operandnya sebagai subpohon, dll.

Binary tree dapat diimplementasikan dalam C++ dengan menggunakan double linkedlist.



Langkah-langkah Pembentukan Binary Tree

1. Siapkan node baru
   a. alokasikan memory-nya
   b. masukkan info-nya
   c. set pointer kiri & kanan = NULL
2. Sisipkan pada posisi yang tepat
   - **penelusuran**→ utk menentukan posisi yang tepat; info yangnilainya lebih besar dari parent akan ditelusuri di sebelah kanan, yang lebih kecil dari parent akan ditelusuri di sebelah kiri
   - **penempatan**→ info yang nilainya lebih dari parent akan ditempatkan di sebelahkanan, yang lebih kecil di sebelah kiri

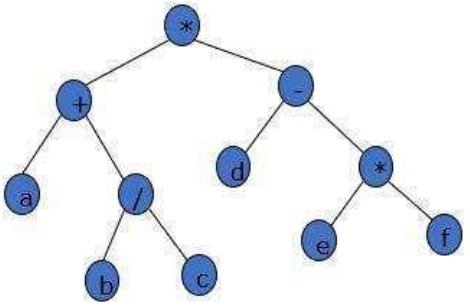
Salah satu operasi yang paling umum dilakukan terhadap sebuah tree adalah kunjungan (traversing)

Ada 3 urutan dasar yang dapat digunakan untuk mengunjungi pohon, yaitu :

1. **PreOrder** : Cetak isi node yang dikunjungi, kunjungi Left Child, kunjungi Right Child.
2. **InOrder** : Kunjungi Left Child, cetak isi node yang dikunjungi, kunjungi Right Child.

3. **PostOrder** : Kunjungi Left Child, kunjungi Right Child cetak isi node yang dikunjungi.

Logical Aritmatika



Hasil :

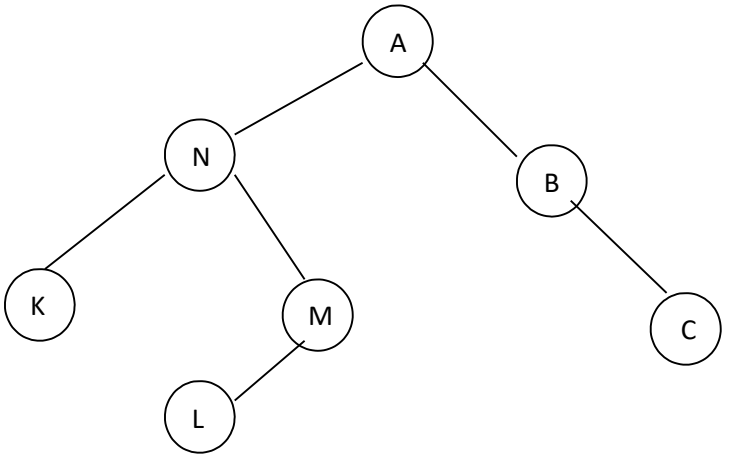
1. PreOrder : *+a/bc-d*ef
2. InOrder : a+b/c*d-e*f
3. PostOrder : abc/+def*-*

Program

```
#include<iostream>
#include<stdio.h>
#include<conio.h>
using namespace std;

struct node
    {
    char data;
    node *kiri;
    node *kanan;
    };

node *akar=NULL;

node *addNode(node **akar, char isi) {
 if((*akar)==NULL){
  node *baru;
  baru= new node;
  baru->data = isi;
  baru->kiri = NULL;
  baru->kanan = NULL;
  (*akar)=baru;
 }
}
node *preOrder(node *akar) {
 if(akar !=NULL) {
  cout<<" "<<akar->data;
  preOrder(akar->kiri);
  preOrder(akar->kanan);
 }
}

node *inOrder(node *akar) {
 if(akar !=NULL) {
  inOrder(akar->kiri);
  cout<<" "<<akar->data;
  inOrder(akar->kanan);
 }
}

node *postOrder(node *akar) {
 if(akar !=NULL) {
```

```
44    postOrder(akar->kiri);
45    postOrder(akar->kanan);
46    cout<<" "<<akar->data;
47  - }
48  }
49
50= main(){
51  char abjad;
52  cout<<"\n\n\tPosisi Awal Tree:\n\n";
53  cout<<"\t       A\n\t      / \\\n\t     B   C\n\t    /\n\t   D\n\t  / \\\n\t E   F\n\n";
54  adcNode(&akar,abjad='A');
55  adcNode(&akar->kiri,abjad='B');
56  adcNode(&akar->kanan,abjad='C');
57  adcNode(&akar->kiri->kiri,abjad='D');
58  adcNode(&akar->kiri->kiri->kiri,abjad='E');
59  adcNode(&akar->kiri->kiri->kanan,abjad='F');
60   cout<< Tampilan PreOrder  : ";
61   preOrder(akar);
62   cout<< \nTampilan InOrder   : ";
63   inOrder(akar);
64   cout<<"\nTampilan PostOrder : ";
65   postOrder(akar);
66 - }
```

1. Buatlah Program C++ Binary Tree dengan 3 Operasi dasar (PreOrder, InOrder dan PostOrder) sesuai gambar dibawah ini :



DAFTAR PUSTAKA

Munir, Rinaldi. (2005). *Algoritma dan Pemrograman Dalam Bahasa Pascal dan C, Edisi III*. Bandung: Informatika Bandung. ISBN: 979-95779-3-4.

Schildt, Herbert. (2003). *C/C++ Programmer's Reference, Third Edition.* California: McGraw-Hill/Osborne. ISBN: 0072227222.