

# BAB I

## STRUKTUR DASAR ALGORITMA

### 1.1 Tujuan

1. Mahasiswa dapat mengetahui dan memahami macam struktur dasar algoritma
2. Mahasiswa dapat mengetahui dan memahami struktur sekuensial
3. Mahasiswa dapat mengetahui dan memahami struktur seleksi
4. Mahasiswa dapat mengetahui dan memahami struktur pengulangan

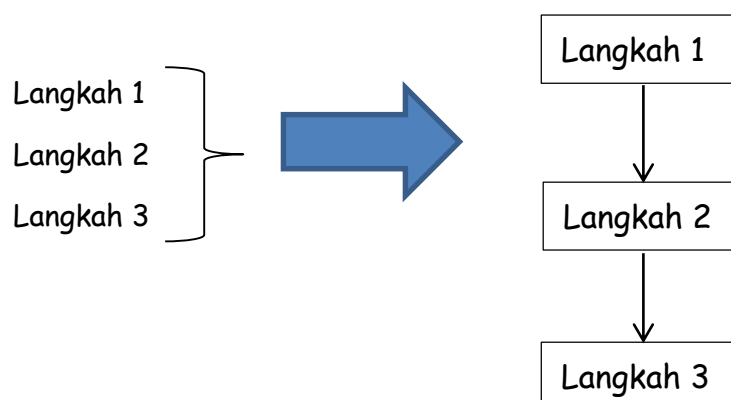
### 1.2 Dasar Teori

#### A. Sekuensial

Suatu pemrograman tidaklah terlepas dari algoritma sebagai acuan untuk membuat program. Pemrograman sekuensial merupakan runtunan (sekuensial / *sequence*) satu atau lebih instruksi, yang berarti bahwa:

- Tiap instruksi dikerjakan satu per satu
- Tiap instruksi dilaksanakan tepat satu kali; tidak ada instruksi yang di ulang
- Urutan instruksi yang dilaksanakan pemroses sama dengan urutan instruksi sebagaimana yang tertulis di dalam teks algoritmanya.
- Akhir dari instruksi terakhir merupakan akhir algoritma.

Pada struktur sekuensial, langkah-langkah yang dilakukan dalam algoritma diproses secara berurutan, sebagaimana diperlihatkan pada gambar berikut:



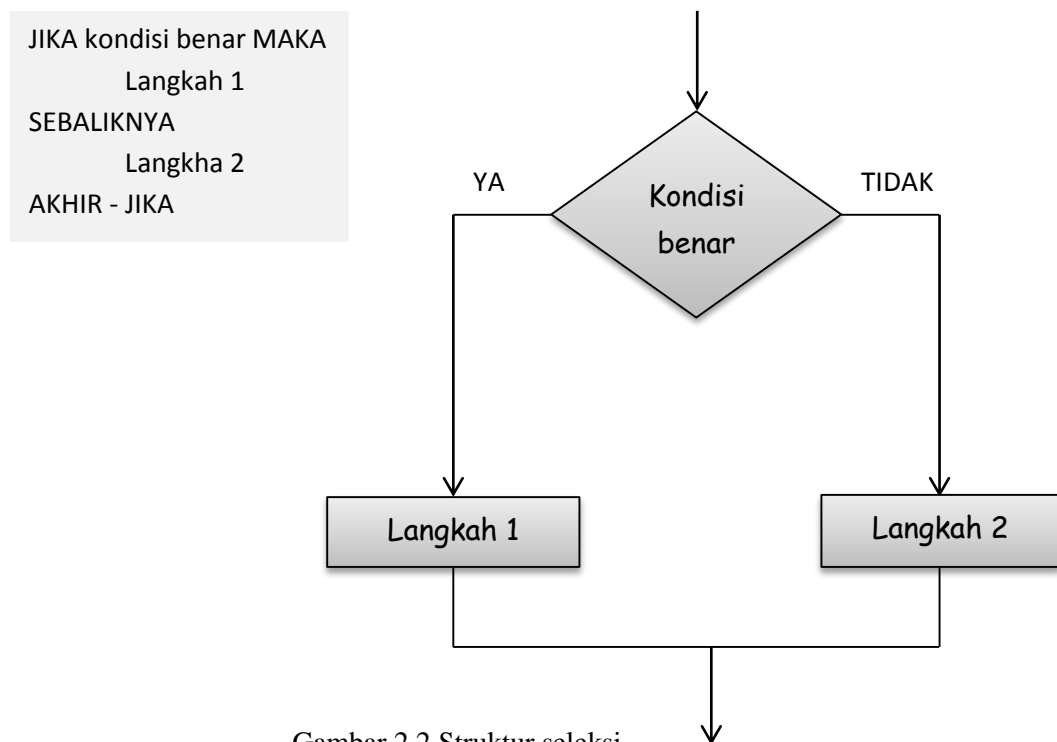
Gambar 2.1 Struktur sekuensial

Pada contoh tersebut, langkah 1 merupakan langkah yang akan dijalankan pertama kali. Setelah itu, langkah 2 dikerjakan dan diikuti dengan langkah 3.

## B. Seleksi

Seleksi digunakan untuk mengarahkan pencabangan aliran eksekusi program. Eksekusi program diarahkan sesuai kondisi yang mengendalikannya. Aliran seleksi dalam C++ dinyatakan dalam bentuk if dan switch. Aliran ini digunakan untuk menentukan alur jalannya eksekusi program. Alur eksekusi program ditentukan dari nilai kebenaran kondisi yang di tentukan.

Struktur seleksi menyatakan pemilihan langkah yang didasarkan oleh suatu kondisi (pengambilan keputusan). Gambar 2.2 memperlihatkan diagram alir struktur seleksi yang melibatkan dua alternatif. Dalam hal ini, simbol belah ketupat digunakan untuk mewakili langkah pengambilan keputusan.



Gambar 2.2 Struktur seleksi

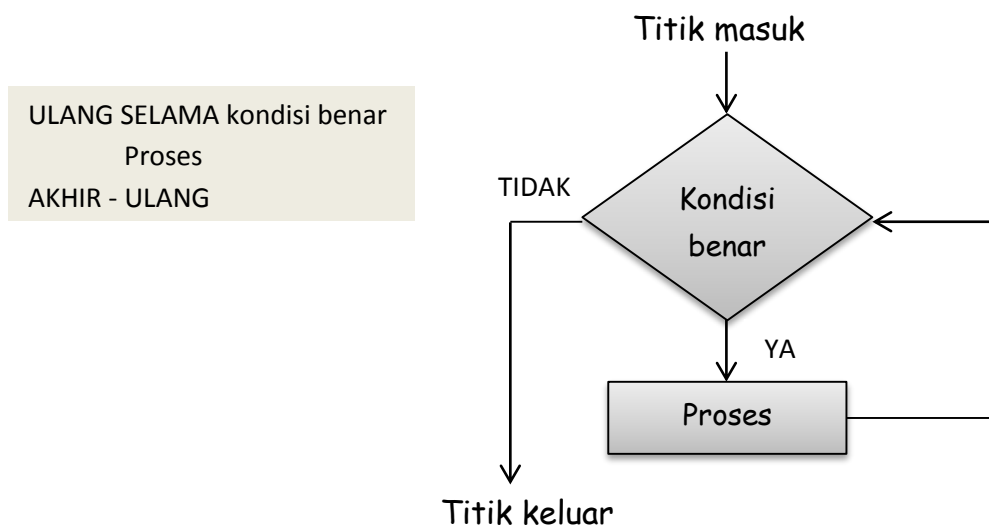
Pada struktur tersebut, langkah 1 hanya akan dijalankan kalau kondisi bernilai benar, sedangkan langkah 2 hanya akan dijalankan kalau kondisi bernilai salah.

### C. Pengulangan

Pengulangan adalah suatu tindakan melakukan hal yang sama berulang-ulang kali. Penting untuk dilakukan suatu pengulangan atau perulangan dalam pembuatan suatu program dengan tujuan untuk mengefisienkan kerja program ataupun mengefisienkan listing dari coding dari program tersebut, karena programmer tidak perlu mengulang proses yang sama. Untuk perulangan pada c++ dapat dilakukan dengan menggunakan pernyataan for, while, do-while.

Pengulangan menyatakan suatu tindakan atau langkah yang dijalankan beberapa kali. Struktur pengulangan menyatakan perwujudan keadaan seperti itu. Sebagai contoh, jika anda ingin menampilkan 10 tulisan “SELAMAT BELAJAR”, Anda bisa menuliskannya dengan menggunakan struktur sekuensial. Hal itu berarti Anda memberikan 10 instruksi untuk menuliskan kesepuluh tulisan tersebut. Cara seperti itu memang praktis untuk jumlah pengulangan yang sedikit (misalnya 2 atau 3 pengulangan), tetapi tidak cocok untuk jumlah yang besar. Agar lebih praktis, Anda bisa menggunakan struktur pengulangan.

#### Struktur pengulangan pertama

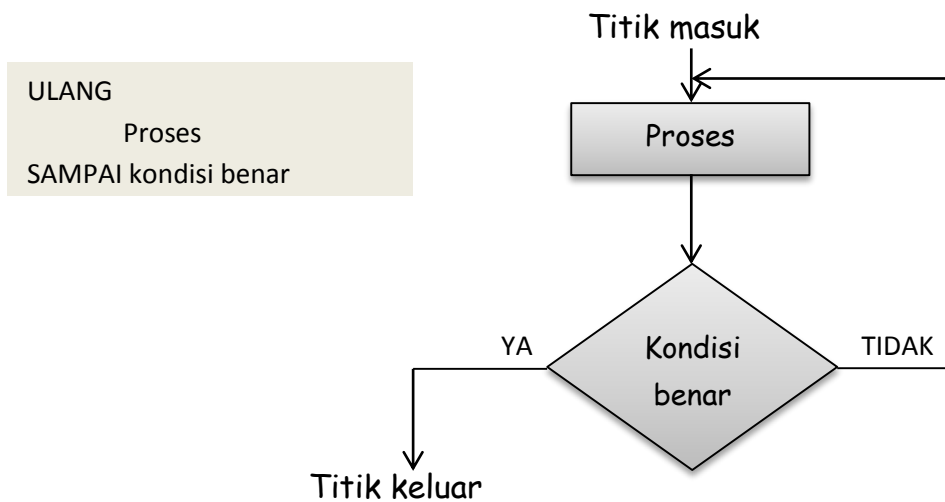


Gambar 2.3 Diagram alir untuk menggambarkan pengulangan bentuk pertama

Pada struktur tersebut, proses dapat berupa satu atau beberapa langkah. Pada bentuk ini, ada kemungkinan proses tidak dijalankan sama sekali sekiranya di awal kondisi bernilai salah. Diagram alir pada Gambar 2.3 menunjukkan bahwa sebelum proses dijalankan pertama kali, kondisi diuji terlebih dahulu. Sekiranya kondisi

bernilai benar maka proses dijalankan, kemudian kondisi diuji lagi. Sepanjang kondisi masih bernilai benar proses akan teta dijalankan. Namun, begitu kondisi bernilai salah maka pengulangan berakhir.

## Struktur pengulangan kedua



Gambar 2.4 Diagram alir untuk menggambarkan pengulangan bentuk kedua

Pada bentuk kedua, proses paling tidak dijalankan sekali.

## 1.3 Latihan

### 1. Contoh seleksi

```
#include <iostream>

Using namespace std;
main()
{

char jenis;
int panjang, lebar, luas, jari;
float luaslingkar;
cout << "Pilih Persegi panjang<P> atau Lingkaran<L>: ";
cin >> jenis;

if (jenis == 'P' || jenis == 'p')
{
    cout << "panjang = ";
    cin >> panjang;
    cout << "lebar = ";
    cin >> lebar;
    luas = panjang * lebar;
```

```

        cout << "Luas = " << luas << endl;
    }
    else if (jenis == 'L' || jenis == 'l')
    {
        cout << "jari-jari = ";
        cin >> jari;
        luaslingkar = 3.14 * jari * jari;
        cout << "Luas = " << luaslingkar << endl;
    }
    else
        cout << "salah pilih" << endl;
}

```

## 2. Contoh Perulangan

```

#include <iostream>

using namespace std;
main()
{
    int l=0;
    for (int j = 0; j <= 10; j++)
    {
        for (int k = j; k <=l; k++)
        {
            cout<<k<<' ';
        }
        l++;
        cout<<"\n";
    }
}

```

Output :

```

0
1 2
2 3 4
3 4 5 6
4 5 6 7 8
5 6 7 8 9 10
6 7 8 9 10 11 12
7 8 9 10 11 12 13 14
8 9 10 11 12 13 14 15 16
9 10 11 12 13 14 15 16 17 18
10 11 12 13 14 15 16 17 18 19 20

```

## 1.4 Tugas

1. Modifikasilah program perulangan pada contoh 2 dengan struktur while menggunakan bahasa C.
2. Buatlah program yang dapat menghitung luas, keliling, dan diagonal persegi panjang. Diagonal dihitung dengan rumus:  
$$\text{diagonal} = \text{akar}(\text{panjang}^2 + \text{lebar}^2).$$
akar dapat dikerjakan dengan fungsi sqrt().  
Masukan berupa panjang dan lebar. Program berupa pilihan untuk luas, keliling, dan diagonal.
3. Modifikasilah program nomor 2 menjadi bahasa java
4. Buatlah program agar hasil outputnya menjadi persegi seperti gambar di bawah ini dalam bahasa C, dengan ketentuan panjang sisinya di inputkan.



5. Buatlah simulasi menu program dengan tampilan di bawah ini menggunakan WHILE.

MENU PILIHAN

1. Dangdut
2. Pop
3. Rock
4. Exit

Pilihan Anda (1/2/3/4) ? ...

Apabila dipilih menu no 1, maka akan tampil teks "Anda memilih menu 1". Demikian pula untuk menu 2 dan 3. Kemudian setelah itu muncul teks "Tekan ENTER untuk kembali ke menu utama". Artinya begitu kita tekan ENTER menu pilihan akan muncul kembali, dst. Akan tetapi bila yang dipilih menu 4 (EXIT), program langsung berhenti.

## **BAB II**

### **ARRAY 2D DAN 3D**

#### **2.1 Tujuan**

1. Mengetahui array 2 dimensi dan array 3 dimensi
2. Mengetahui dan memahami penggunaan array 2 dimensi dan 3 dimensi.

#### **2.2 Dasar Teori**

##### **A. Pengertian Array**

Array adalah sebuah variable yang menyimpan sekumpulan data yang memiliki tipe yang sama. Setiap data tersebut menempati lokasi atau alamat memori yang berbeda – beda dan selanjutnya disebut dengan elemen array. Elemen array itu kemudian dapat kita akses melalui indeks yang terdapat didalamnya.

Untuk mendeklarasikan sebuah array dalam C++, kita harus menggunakan tanda [] (*bracket*).

##### **B. Array Multidimensi**

Array Multidimensi yaitu array yang terdiri dari beberapa subskrip array. Sebagai contoh, array 2 dimensi adalah array yang mempunyai 2 subskrip array, 3 dimensi mempunyai 3 subskrip array dan seterusnya. Array seperti ini sering digunakan untuk pemrosesan matrik.

##### **1. Array Dua Dimensi**

Array dua dimensi adalah array yang memiliki dua buah elemen bertipe array. Dengan kata lain, array dua dimensi memiliki dua buah subskrip, yang biasanya dipresentasikan dengan baris dan kolom.

Contoh sederhana adalah data yang tertulis dalam tabel berikut ini :

Merk Hp	1992	1993	1994	1995
1. Nokia	35	45	80	120
2. Samsung	100	110	70	101
3. SONY	10	15	20	17

Jika dibuat programnya adalah sebagai berikut :

```
int data_hp [3] [4];
```

Bentuk umum pendeklarasian sebuah array dua dimensi adalah sebagai berikut :

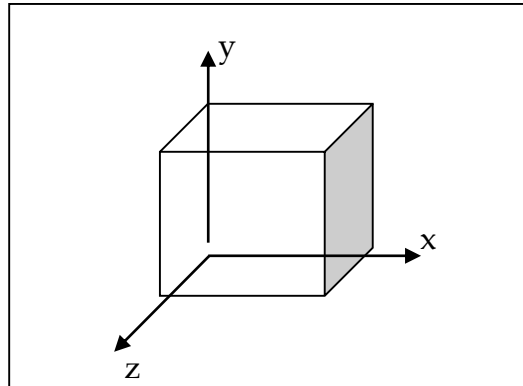
**tipe\_data**

**nama\_array[jumlah\_elemen\_baris][jumlah\_elemen\_kolom];**

## 2. Array Tiga Dimensi

Array tiga dimensi merupakan array yang memiliki tiga buah subskrip dan lebih kompleks apabila dibandingkan dengan array dua dimensi. Di sini, subskrip yang ada akan dipresentasikan dengan sumbu x, y dan z atau panjang, lebar dan tinggi seperti yang ditunjukkan oleh gambar berikut.





Bentuk umum pendeklarasian sebuah array tiga dimensi adalah sebagai berikut :

```
tipe_data  
nama_array[jumlah_elemen_x][jumlah_elemen_y][  
    jumlah_elemen_z];
```

## 2.3 Latihan

### 1. Latihan 1

```
#include <iostream>
using namespace std;
main()
{
    int i, j;
    int data[2][2];

    cout<<"Input Data"<<endl;
    for(i=0;i<=1;i++)
    {
        for(j=0;j<=1;j++)
        {
            cout<<"data["<<i+1<<"] ["<<j+1<<"]="";
            cin >> data[i][j];
        }
    }
    cout<<"Matrik yang diinputkan : "<<endl;
    for(i=0;i<=1;i++)
    {
        for(j=0;j<=1;j++)
        {
            cout<<data[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

## 2. Latihan 2

```
#include <iostream>

using namespace std;

int main()
{
    float nilai[2][3];
    int baris, kolom;

    for(baris=0; baris<2; baris++)
    {
        for(kolom=0; kolom<3; kolom++)
        {
            cout<<"Nilai [baris][kolom] ke-
["<<baris+1<<"] ["<<kolom+1<<"] = ";
            cin>>nilai[baris][kolom];
        }
    }
    cout<<endl;
    cout<<"Matriks Nilai yang dimasukkan :
"<<endl<<endl;
    for(baris=0; baris<2; baris++)
    {
        for(kolom=0; kolom<3; kolom++)
        {
            cout<<nilai[baris][kolom]<<" ";
        }
        cout<<endl;
    }
}
```

```

}
cout<<endl;

float jml_nilai1, jml_nilai2;
float rerata1, rerata2;
jml_nilai1=0;
jml_nilai2=0;

float jml_nilaicol1, jml_nilaicol2, jml_nilaicol3;
float reratacol1, reratacol2, reratacol3;
jml_nilaicol1=0;
jml_nilaicol2=0;
jml_nilaicol3=0;

for (baris=0; baris<2; baris++)
{
    if (baris==0)
    {
        for(kolom=0; kolom<3; kolom++)
            jml_nilai1=jml_nilai1 + nilai[baris][kolom];
    }
    else
    {
        for (kolom=0; kolom<=3; kolom++)
            jml_nilai2=jml_nilai2 + nilai[baris][kolom];
    }
}

rerata1=jml_nilai1/3;

```

```

rerata2=jml_nilai2/3;

cout<<"Rerata baris ke 1 = "<<rerata1<<endl;
cout<<"Rerata baris ke 2 = "<<rerata2<<endl;

//per kolom
cout<<endl;
for(kolom=0; kolom<3; kolom++)
{
    if(kolom==0)
    {
        for(baris=0; baris<2; baris++)
            jml_nilaicol1=jml_nilaicol1 +
nilai[baris][kolom];
    }
    else
    if (kolom==1)
    {
        for (baris=0; baris<2; baris++)
            jml_nilaicol2=jml_nilaicol2 +
nilai[baris][kolom];
    }
    else
    {
        for(baris=0; baris<2; baris++)
            jml_nilaicol3=jml_nilaicol3 +
nilai[baris][kolom];
    }
}

```

```
reratacol1=jml_nilaicol1/2;
reratacol2=jml_nilaicol2/2;
reratacol3=jml_nilaicol3/2;

cout<<"Rerata kolom ke 1 = "<<reratacol1<<endl;
cout<<"Rerata kolom ke 2 = "<<reratacol2<<endl;
cout<<"Rerata kolom ke 3 = "<<reratacol3<<endl;

}
```

### 3. Latihan 3

```
#include <iostream>

using namespace std;

main()
{
    int i, j, k, x=1;
    int matrik[3][3][4];

    cout<<" 3 dimensi array "<<endl<<endl;

    for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
        {
```

```

        for (k=0;k<5;k++)
        {
            matrik[i][j][k] = x;
            cout<<" "<<matrik[i][j][k];
            x++;
        }
        cout<<endl;
    }
    cout<<endl;
}
}

```

## 2.4 TUGAS

1. Ubahlah latihan 2 kedalam bahasa java!
2. Jalankan program berikut :

```

#include <iostream>

using namespace std;

int main()
{
    typedef int matrik[3][2];
    hitung elemen1, elemen2, hasil_jumlah;

    int baris, kolom;

```

```

for (baris=1; baris<=3; baris++)
{
    for (kolom=1; kolom<=2; kolom++)
    {
        cout<<"Elemen
A["<<baris<<"] ["<<kolom<<"] = ";
        cin>>elemen1 [baris] [kolom];
    }
}
cout<<endl;
for (baris=1; baris<=3; baris++)
{
    for (kolom=1; kolom<=2; kolom++)
    {
        cout<<"Elemen
B["<<baris<<"] ["<<kolom<<"] = ";
        cin>>elemen2 [baris] [kolom];
    }
}

cout<<endl;
{
    for (baris=1; baris<=3; baris++)
    {
        for (kolom=1; kolom<=2; kolom++)
        {

hasil_jumlah [baris] [kolom]=elemen1 [baris] [kolo
m]+elemen2 [baris] [kolom];

```



```

    }

}

clrscr();

cout<<"Hasil Penjumlahan Elemen A dan Elemen
B adalah : "<<endl;
for(baris=1; baris<=3; baris++)
{
    for(kolom=1; kolom<=2; kolom++)
    {
        cout<<"Hasil Jumlah ["<<baris<<"]["<<kolom<<"] =
"<<hasil_jumlah[baris][kolom]<<endl;
    }
}
}

```

3. Modifikasi tugas 2 dengan menginputkan jumlah baris dan jumlah kolom. Kemudian tambahkan operasi pengurangan dan perkalian didalamnya !
4. Jalankan program berikut dan ganti kedalam bahasa C++ :

```

#include <stdio.h>

using namespace std;

main()
{
    char h=70, nama[4][3][13] =
    {
        "Angsa", "Anjing", "Anoa",
        "Banteng", "Badak", "Beruang",
    }
}

```

```

        "Camar", "Capung", "Cendrawasih",
        "Dara", "Domba", "Duyung",

    };

    printf("Daftar Hewan sesuai abjad : \n\n");





    for(int i=0; i<4; i++)
    {
        ++h;
        printf("Kelompok %c \n", h);
        for(int s=1; s<=3; s++)
        {
            printf("%d. %s \n", s,
nama[i][s]);
        }
        printf("\n");
    }
}

```

5. Buatlah program untuk membentuk sebuah kelompok dimana jumlah kelompok dan jumlah anggota diinputkan. Kemudian tampilkan kembali kelompok dan anggota yang telah diinputkan seperti gambar berikut :

Output

array3d\_buat\_kelompok (Build, Run)array3d\_buat\_kelompok (Run)

kelompok 1  
1. Andre  
2. yusuf  
3. ismail  
  
kelompok 2  
1. irna  
2. satri  
3. milsa  
  
kelompok 3  
1. kurnia  
2. hendy  
3. agus  
  
RUN SUCCESSFUL (total time: 1m 10s)

## **MODUL III**

### **METODE SORTING DAN SEARCHING**

#### **1. Tujuan Praktikum.**

- a. Mengerti dan Memahami Metode Pencarian Data.
- b. Dapat menggunakan Metode Pencarian Data dalam Program.
- c. Memahami penggunaan metode Sorting.

#### **2. Dasar Teori.**

##### **a. Pencarian Data.**

Pencarian (searching) merupakan tindakan untuk mendapatkan suatu data dalam kumpulan data. Dalam kehidupan sehari-hari, seringkali kita berurusan dengan pencarian, misalnya untuk menemukan nomor telepon seseorang pada buku telepon atau mencari suatu istilah dalam kamus.

Terdapat beragam algoritma pencarian untuk keperluan mencari data. Perlu diketahui, yang dimaksud dengan algoritma pencarian adalah “algoritma yang menerima sebuah argument *a* dan mencoba untuk menemukan sebuah rekaman yang memiliki kunci *a*”.

Pencarian data dilakukan terhadap data dapat dilakukan terhadap data yang secara keseluruhan berada dalam memory computer ataupun terhadap data yang berada dalam penyimpanan eksternal.

##### **b. Bubble Sort.**

Sorting merupakan suatu proses untuk menyusun kembali himpunan obyek menggunakan aturan tertentu. Sorting disebut juga sebagai suatu algoritma untuk meletakkan kumpulan elemen data kedalam urutan tertentu berdasarkan satu atau beberapa kunci dalam tiap-tiap elemen. Pada dasarnya ada dua macam urutan yang biasa digunakan dalam suatu proses sorting:

##### **1. Urut naik (ascending)**

Mengurutkan dari data yang mempunyai nilai paling kecil sampai paling besar

## 2. Urut turun (descending)

Mengurutkan dari data yang mempunyai nilai paling besar sampai paling kecil.

Metode-metode sorting yang akan dibahas kali ini meliputi:

1. Bubble sort (Metode Gelembung)
2. Insertion Sort (Metode Penyisipan) / Selection Sort.
3. Selection Sort (Metode Seleksi)
4. Quick Sort (Metode Quick)

Metode gelembung (bubble sort) sering juga disebut dengan metode penukaran (exchange sort) adalah metode yang mengurutkan data dengan cara membandingkan masing-masing elemen, kemudian melakukan penukaran bila perlu. Metode ini mudah dipahami dan diprogram, tetapi bila dibandingkan dengan metode lain yang kita pelajari, metode ini merupakan metode yang paling tidak efisien.

### **Algoritma Bubble Sort**

1. Membandingkan data ke- $i$  dengan data ke- $(i+1)$  (tepat bersebelahan). Jika tidak sesuai maka tukar (data ke- $i$  = data ke- $(i+1)$  dan data ke- $(i+1)$  = data ke- $i$ ). Apa maksudnya tidak sesuai? Jika kita menginginkan algoritme menghasilkan data dengan urutan ascending (A-Z) kondisi tidak sesuai adalah data ke- $i >$  data ke- $i+1$ , dan sebaliknya untuk urutan descending (A-Z).
2. Membandingkan data ke- $(i+1)$  dengan data ke- $(i+2)$ . Kita melakukan perbandingan ini sampai data terakhir. Contoh: 1 dgn 2; 2 dgn 3; 3 dgn 4; 4 dgn 5 ... ;  $n-1$  dgn  $n$ .

3. Selesai satu iterasi, adalah jika kita sudah selesai membandingkan antara (n-1) dgn n. Setelah selesai satu iterasi kita lanjutkan lagi iterasi berikutnya sesuai dengan aturan ke-1. mulai dari data ke-1 dgn data ke-2, dst.
4. Proses akan berhenti jika tidak ada pertukaran dalam satu iterasi.

Latihan.

1. Latihan 1.

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

int cari (int data[],int n,int k)
{
    int posisi,i,ketemu;
    if (n <= 0)
        posisi=-1;
    else{
        ketemu = 0;
        i=1;
        while ((i<n-1)&&! ketemu)
            if (data[i] == k)
            {
                posisi=i;
                ketemu=1;
            }
        else
            i++;
        if (!ketemu)
            posisi = -1;
    }
    return posisi;
}

int main()
{
    int data [8];
    int dicari;
```

```

    cout <<"Masukkan Data : " ;
    for(int a=0;a<8;a++)
    {
        cin >> data[a];
    }
    cout << "Masukkan Data yang dicari : ";
    cin >>dicari;
    cout<<"Posisi " <<dicari<<" dalam larik data: "
    <<(cari(data,8,dicari))<<"\n";
    getch();
}

```

## 2. Latihan 2.

### Contoh Program Bubble Sort.

```

#include <iostream>
using namespace std;
int main()
{
    int i,j,n,data[10],simpan,k;
    cout<<"masukkan banyak data= ";cin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"data "<<i<<" = ";cin>>data[i];
    }
    cout<<"awal = ";
    for(i=1;i<=n;i++)
    cout<<data[i]<<" ";
    cout<<endl;

    for(i=1;i<n;i++)
    {
        for(j=1;j<n;j++)
        {
            if(data[j]>data[j+1])
            {
                simpan=data[j];
                data[j]=data[j+1];
                data[j+1]=simpan;
            }
        }
    }
    cout<<"hasil= ";
}

```

```

        for(i=1;i<=n;i++)
            cout<<data[i]<<" ";
    return 0;
}

```

**Tugas.**

### 1. Tugas 1.

Ubahlah script di bawah ini menggunakan bahasa C.

```

#include <iostream>
using namespace std;
int data[10],data2[10];
int n;
void tukar(int a,int b)
{
    int t;
    t = data[b];
    data[b] = data[a];
    data[a] = t;
}
void Input()
{
    cout<<"Masukkan jumlah data = ";cin>>n;
    cout<<"-----"<<endl;
    for(int i=0;i<n;i++)
    {
        cout<<"Masukkan data ke-"<<(i+1)<<" = ";cin>>data[i];
        data2[i] = data[i];
    }
    cout<<endl;
}
void Tampil()
{
    for(int i=0;i<n;i++)
    {
        cout<<data[i]<<" ";
    }
    cout<<endl;
}void bubble_sort()
{
    for(int i=1;i<n;i++)
    {
        for(int j=n-1;j>=i;j--)

```



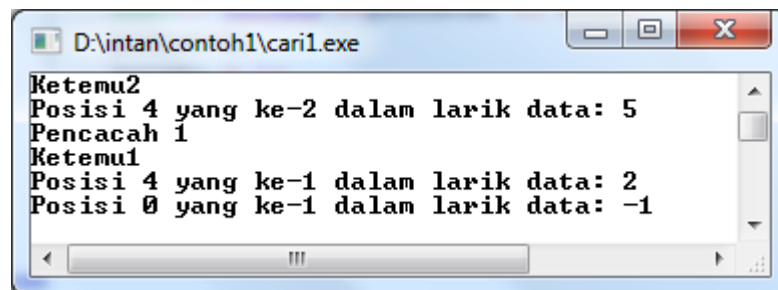
```

{
if(data[j]<data[j-1])  tukar(j,j-1);
}
Tampil();
}
cout<<endl;
}
main()
{
cout<<"* Bubble Sort *"<<endl;
Input();
cout<<"Proses Bubble Sort,,,,,,,,,"<<endl;
cout<<"-----"<<endl;
Tampil();
bubble_sort();
cout<<"-----"<<endl;}

```

## 2. Tugas 2.

Jalankan program agar menghasilkan tampilan seperti gambar berikut:



```

#include <iostream>
using namespace std;
int cari (int data[],int n,int k,int m)
{
    int posisi,i,ketemu,pencacah=0;
    if (n <= 0)
        posisi=-1;
    else{
        ketemu = 0;
        i=1;
        while ((i<n+1)&&! ketemu)
            if (data[i] == k)

```

```

        {
            pencacah++;
            cout<<"Pencacah " <<pencacah <<endl;
            if (pencacah != m)
            {
                cout<<"Ketemu"<<pencacah << endl;
                posisi=i;
                ketemu = 1;
            }
            else
                I--;

        }
        else
            i++;
        if (!ketemu)
        }
        return posisi;
    }
}

int main()
{
    int data [9]={10,9,4,6,3,4,2,5,0};
    int dicari;
    int ke;

    dicari=4;
    ke =2;
    cout<<"Posisi " <<dicari<<" yang ke-"<<ke<<" dalam
    larik data: " <<(cari(data,8,dicari,ke))<<"\n";
    ke =1;
    cout<<"Posisi " <<dicari<<" yang ke-"<<ke<<" dalam
    larik data: " <<(cari(data,8,dicari,ke))<<"\n";
    dicari = 0;
    ke=1;
    cout<<"Posisi " <<dicari<<" yang ke-"<<ke<<" dalam
    larik data: " <<(cari(data,9,dicari,ke))<<"\n";
    return 0;
}

```

Tugas 3.

Ubahlah Script pada latihan 1 dengan menggunakan metode Insertion Sort secara descending.

### 3. Tugas 4.

Buatlah program pengurutan data dengan menggunakan semua metode pengurutan untuk mengurutkan angka secara ascending dan descending.

### 4. Tugas 5.

Ubahlah script di bawah ini dengan menggunakan bahasa java.

```
#include <stdio.h>
void tampilkan_larik (int data[],int n)
{
    int i;
    for(i=0;i< n ;i++)
        printf("%d ",data[i]);
    printf("\n");
}

int partisi(int data[],int p,int r)
{
    int pivot= data[r];
    int i = p-1;
    int j;
    for (j=p;j<r;j++)
    {
        if (data[j]<= pivot)
        {
            i++;
            int tmp = data[i];
            data[i]= data[j];
            data[j]= tmp;
        }
    }
    int tmp = data[i+1];
    data[i+1]= data[r];
    data[r]= tmp;
    return i+1;
}

void quick_sort(int data[],int p,int r)
{

```

```

int q;
if (p<r)
{
    q = partisi(data,p,r);
    quick_sort(data,p,q-1);
    quick_sort(data,q+1,r);

}
}

int main()
{
    int jum_data = 9;
    int i;
    int data[]= {25,57,48,37,12,92,80,33,1};

    quick_sort(data,0,jum_data-1);
    printf("hasil pengurutan : \n");
    tampilkan_larik(data,jum_data);
    return 0;
}

```

## REKURSI

rekursi adalah suatu kemampuan subrutin untuk memanggil dirinya sendiri. Adapun suatu subrutin yang memanggil dirinya seperti itu dinamakan subrutin rekursi. Pada beberapa persoalan, kemampuan seperti itu sangat berguna karena mempermudah solusi. Namun demikian rekursi juga memiliki kelemahan, yakni memungkinkan terjadinya *overflow* pada *stack* (*stack* tidak lagi mampu menangani permintaan pemanggilan subrutin karena kehabisan memori). Itu sebabnya harus ada jaminan bahwa proses rekursi akan berhenti pada suatu waktu tertentu, yang menyebabkan pemanggilan fungsi berakhir.

Contoh 1 : Tuliskan algoritma untuk menghitung nilai suatu faktorial  $n$  beserta programnya.

Algoritma:

```
SUBROUTIN faktorial(n)
    JIKA  $n = 0$  ATAU  $1$  MAKA
        NILAI-BALIK  $1$ 
    SEBALIKNYA
        NILAI-BALIK  $n \times \text{faktorial}(n-1)$ 
    AKHIR-JIKA
AKHIR-SUBROUTIN
```

**Program:**

Berikut ini merupakan implementasi program dalam bahasa C.

```
#include <stdio.h>

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

int main()
{
    int n;
    long int hasil;

    printf("n = ");
    scanf("%d", &n);

    hasil = faktorial(n);
    printf("%d! = %ld", n, hasil);

    return 0;
}
```

Contoh 2 : Tuliskan algoritma untuk menampilkan nilai satu deret fibonaci tertentu beserta programnya.

Algoritama:

```
SUBROUTIN fib(n)
    JIKA n = 0 MAKA
        NILAI-BALIK 0
    SEBALIKNYA JIKA n = 1
        NILAI-BALIK 1
    SEBALIKNYA
        NILAI-BALIK  $fib(n-1) + fib(n-2)$ 
    AKHIR-JIKA
AKHIR-SUBROUTIN
```

Program:

Berikut ini merupakan implementasi program dalam bahasa C++.

```
#include <iostream>
long int fib (unsigned int n)
{
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fib(n-1) + fib(n-2);
}

int main()
{
    int n;
    long int hasil;
    cout<<"n = ";
    cin>>n;
    hasil = fib(n);
    cout<<"fibonaci("<<n<<" ) = "<<hasil;

    return 0;
}
```

## Tugas

1. Berikut ini merupakan algoritma untuk menghitung nilai  $Y^n$  dengan  $n$  berupa bilangan bulat lebih besar dari pada nol yang dihitung secara rekursif.

SUBROUTIN *pangkat(y,n)*

    JIKA  $n = 1$  MAKA

        NILAI BALIK  $y$

    SEBALIKNYA

        NILAI BALIK  $y \times \text{pangkat}(y,n-1)$

    AKHIR JIKA

AKHIR SUBROUTIN

Buatlah program menggunakan bahasa C/C++ untuk menghitung nilai  $Y$  pangkat  $n$  tersebut berdasarkan algoritma di atas.

2. Ubahlah program yang telah dibuat di nomor 1 kedalam bahasa Java.
3. Buatlah algoritma dari subrutin untuk membalik suatu bilangan dengan cara rekursi. Sebagai contoh, bilangan 1261 ditampilkan menjadi 1621.
4. Buatlah program menggunakan bahasa C/C++ berdasarkan algoritma pada soal nomor 3.
5. Ubahlah program yang telah dibuat pada soal nomor 4 ke dalam bahasa Java.



## MODUL 5

### LINKED LIST, STACK DAN QUEUE

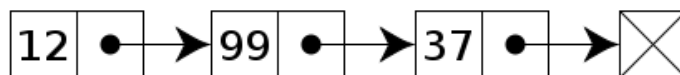
#### 1.1 Tujuan

- Mengetahui penggunaan metode LINKED LIST
- Mengetahui penggunaan metode STACK
- Mengetahui penggunaan metode QUEUE

#### 1.2 Dasar Teori

##### METODE LIST

List atau senarai adalah sebuah pemikiran/ konsep struktur data yang sangat dasar pada pemrograman agar lebih fleksibel, dimana setiap elemen akan ditambahkan saat dibutuhkan. struktur data ini digunakan untuk menyimpan sejumlah objek data biasanya secara terurut sehingga memungkinkan penambahan, pengurangan, dan pencarian atas elemen data yang tersimpan dalam senarai dilakukan secara lebih efektif.

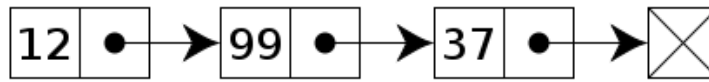


*Sebuah senarai berantai dengan tiap-tiap node yang terdiri atas dua elemen, data integer, dan elemen rujukan ke node berikutnya*

Jenis – jenis Senarai berantai ada 3:

##### 1. Senarai Tunggal

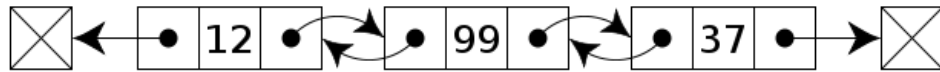
Bila struktur data sebuah node hanya memiliki satu tautan atas node berikutnya dalam sebuah senarai, maka senarai tersebut dinamakan sebagai senarai tunggal.



*Senarai tunggal dengan tiap-tiap node yang terdiri atas dua elemen, data integer, dan elemen rujukan ke node berikutnya*

## 2. Senarai Ganda

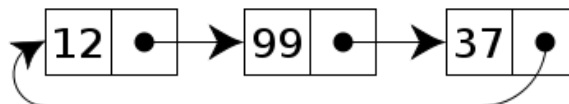
Berbeda halnya dengan senarai tunggal, pada senarai ganda, struktur data atas tiap-tiap node memiliki rujukan pada node sebelum dan berikutnya. Sebagian algoritma membutuhkan taut ganda, contohnya sorting dan reverse traversing.



*Senarai ganda dengan tiap-tiap node yang terdiri atas tiga elemen, data integer, dan dua elemen rujukan ke node sebelum serta berikutnya*

## 3. Senarai Sirkular

Pada dua jenis senarai sebelumnya, node terakhir dalam senarai tersebut merujuk pada **null** yang artinya akhir dari sebuah senarai, begitu pula **null** sebagai rujukan node sebelumnya pada node pertama bila senarai yang dimaksudkan adalah senarai ganda. Pada senarai sirkular, informasi rujukan pada node terakhir akan merujuk pada node pertama, dan rujukan pada node pertama akan merujuk pada node terakhir bila yang digunakan sebagai dasar implementasi adalah senarai ganda.



*Senarai sirkular dengan menggunakan model implementasi senarai tunggal. Node terakhir menyimpan rujukan pada node pertama*

## METODE STACK

sebuah koleksi objek yang menggunakan prinsip LIFO (Last In First Out), yaitu data yang terakhir kali dimasukkan akan pertama kali keluar dari tumpukan

tersebut. Tumpukan dapat diimplementasikan sebagai representasi berkait atau kontigu (dengan tabel fix). Ciri tumpukan:

- TOP merupakan sebutan untuk elemen paling atas dari suatu stack
- Elemen TOP merupakan elemen yang paling akhir ditambahkan
- Elemen TOP diketahui
- penambahan dan penghapusan elemen selalu dilakukan di TOP
- LIFO

Pemanfaatan tumpukan:

- Perhitungan ekspresi aritmatika (posfix)
- algoritma backtracking (runut balik)
- algoritma rekursif

## **METODE QUEUE**

Queue/antrian adalah ordered list dengan penyisipan di satu ujung, sedang penghapusan di ujung lain. Ujung penyisipan biasa disebut rear/tail, sedang ujung penghapusan disebut front/head. Fenomena yang muncul adalah elemen yang lebih dulu disisipkan akan juga lebih dulu diambil. Queue berdisiplin FIFO (First In, First Out). Queue merupakan kasus khusus ordered list. Dengan karakteristik terbatas itu maka kita dapat melakukan optimasi representasi ADT Queue untuk memperoleh kerja paling optimal.

### **1.3 Latihan**

#### **a. Latihan 1**

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

```

using namespace std;

typedef struct{
    char nim[10];
    char nama[50];
    float nilai;
}nilaiMatKul;

typedef struct{
    int first;
    int last;
    nilaiMatKul data[10];
}queue;

void createEmpty(queue*Q){
    (*Q).first = -1;
    (*Q).last = -1;
}

int isEmpty(queue Q){
    int hasil = 0;
    if(Q.first == -1){
        hasil=1;
    }
    return hasil;
}

int isFull(queue Q){
    int hasil = 0;
    if(Q.last==9){

```

```

        hasil=1;
    }
    return hasil;
}

void add(char nim[], char nama[],float nilai,
queue *Q){
    if(isEmpty(*Q) == 1 ){
        (*Q).first=0;
        (*Q).last=0;
        strcpy((*Q).data[0].nim,nim);
        strcpy((*Q).data[0].nama,nama);
        (*Q).data[0].nilai=nilai;

    }
    else{
        if(isFull(*Q) !=1){
            (*Q).last=(*Q).last+1;
            strcpy((*Q).data[(*Q).last].nim,nim);
            strcpy((*Q).data[(*Q).last].nama,nama);
            (*Q).data[(*Q).last].nilai = nilai;
        }
        else{
            printf("queue penuh\n");
        }
    }
}

void del(queue *Q){
    if((*Q).last ==0){

```

```

        (*Q).first ==-1;
        (*Q).last ==-1;
    }
else{
    int i;
    for(i=( *Q ).first+1 ; i<= (*Q ).last ; i++){
        strcpy(( *Q ).data[i-
1].nim, (*Q ).data[i].nim);
        strcpy(( *Q ).data[i-
1].nama, (*Q ).data[i].nama);
        (*Q ).data[i-1].nilai=(*Q ).data[i].nilai;
    }
    (*Q ).last=(*Q ).last-1;
}
}

void printfQueue(queue Q){
if (Q.first != -1){
    printf("-----isi queue-----\n");
    int i;
    for(i=Q.last; i>=Q.first; i--){
        printf("elemen ke : %d\n", i);
        printf("nim : %s\n", Q.data[i].nim);
        printf("nama: %s\n", Q.data[i].nama);
        printf("nilai: %f\n", Q.data[i].nilai);
    }
    printf("-----\n");
}
else{

```

```

        printf("queue kosong\n");
    }
}

int main(){
    queue Q;
    createEmpty(&Q);
    printfQueue(Q);

    printf("-----\n");
    add("13507701", "Rangga", "64.67", &Q);
    add("13507702", "Anggy", "75.11", &Q);
    add("13507703", "Intan", "84.63", &Q);
    printfQueue(Q);

    printf("-----\n");
    del(&Q);
    del(&Q);
    printfQueue(Q);
    printf("-----\n");
    return 0;
}

```

## **b. Latihan 2**

```

#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

struct DAT
{
    int id;

```

```

        char fname[20];
        char mname[20];
        char lname[20];
        char address[80];
        double salary;
        char tele_no[15];
};

struct NODE
{
    DAT data;
    NODE *N;
    NODE*P;
    NODE(const int i , const char *f, const char *m, const char
    *l, const char *ad, const double s, const char *tel)
    {
        data.id = i;
        strcpy(data.fname,f);
        strcpy(data.mname,m);
        strcpy(data.lname,l);
        strcpy(data.address,ad);
        data.salary = s;
        strcpy(data.tele_no,tel);
        N = NULL;
        P = NULL;
    }
};

class StackLinkedList
{
private:
    NODE *front;
public:
    StackLinkedList(){front = NULL;}
    ~StackLinkedList(){destroyList();}
    void push(NODE *);
    NODE* pop();
    void destroyList();
};

void StackLinkedList::push(NODE *n)
{
    if(front == NULL)
    {
        front = n;
    }
    else
    {
        front->P = n;
        n->N = front;
        front = n;
    }
}

```



```

}

NODE* StackLinkedList::pop()
{
    NODE *temp;
    if( front == NULL )//no nodes
        return NULL;
    else if(front->N == NULL)//there is only one node
    {
        NODE * temp2 = front;
        temp = temp2;
        front = NULL;
        delete temp2;
        return temp;
    }
    else//there are more than one node
    {
        NODE * temp2 = front;
        temp = temp2;
        front = front->N;
        front->P = NULL;
        delete temp2;
        return temp;
    }
}

void StackLinkedList::destroyList()
{
    while(front != NULL)
    {
        NODE *temp = front;
        front = front->N;
        delete temp;
    }
}

void disp(NODE *N)
{
    if( N == NULL )
    {
        cout << "\nStack is Empty!!!" << endl;
    }
    else
    {
        cout << "\nId No.      : " << N->data.id << " ";
        cout << "\nFull Name  : " << N->data.fname << " ";
        cout << N->data.mname << " ";
        cout << N->data.lname << endl;
        cout << "Address      : " << N->data.address << endl;
        cout << "Salary        : " << setprecision(15) << N->data.salary << endl;
    }
}

```

```

        cout << "Tele_no      : " << N->data.tele_no<< endl <<
endl;
    }
}

int main()
{
    StackLinkedList *Stack = new StackLinkedList();

    NODE No1(101,"Anggy","Eka","P","Semarang
120",7851243.9475,"07502334121");
    NODE No2(102,"Wa Ode","Kanartia","Ningsi","Timor leste
121",5681125.9457,"07507534127");
    NODE No3(103,"Fatimah","Nurul","Intan","Irian Jaya
123",2344003.48345,"075078654129");

    Stack->push(&No1);
    Stack->push(&No2);
    Stack->push(&No3);

    disp(Stack->pop());

    disp(Stack->pop());

    disp(Stack->pop());

    disp(Stack->pop());

    delete Stack;
    return 0;
}

```

## 1.4 Tugas

1. Benarkan listing program sehingga dapat dicompile !

```

#include <iostream>
using namespace std
class node {
    int data;
    Node* next;

```

```

public:
    Node() {};
    void SetData(int aData) { data = aData; };
    void SetNext(Node* aNext) { next = aNext; };
    int Data() { return data; };
    Node* Next() { return next; }
};

```

```

class List {
    Node *head;
public:
    List() { head = NULL; };
    void Print();
    void Append(int data);
    void Delete(int data);
};

```

```

void List::Print() {
    Node *tmp = head;
    if ( tmp == NULL ) {
        cout << "NULL" << endl;
        return;
    }
    if ( tmp->Next() == NULL ) {
        cout << tmp->Data();
        cout << " --> ";
    }
}

```

```

        cout << "EMPTY" << endl;
    }
    else {
        do {
            cout << tmp->Data();
            cout << " --> ";
            tmp = tmp->Next();
        }
        while ( tmp != NULL );

        cout << "NULL" << endl;
    }
}

void List::Append(int data) {
    if(data != 300) {
        Node* newNode = new Node();
        newNode->SetData(data);
        newNode->SetNext(NULL);

        Node *tmp = head;
        if ( tmp != NULL ) {
            while ( tmp->Next() != NULL ) {
                tmp = tmp->Next();
            }
            tmp->SetNext(newNode);
        }
        else {
            head = newNode;

```

```
    }  
    }  
}
```

```
void List::Delete(int data) {  
  
    if(data != 300) {  
        Node *tmp = head;  
        if ( tmp == NULL )  
            return;  
        if ( tmp->Next() == NULL ) {  
            delete tmp;  
            head = NULL;  
        }  
        else {  
            Node *prev;  
            do {  
                if ( tmp->Data() == data ) break;  
                prev = tmp;  
                tmp = tmp->Next();  
            } while ( tmp != NULL );  
  
            // Adjust the pointers  
            prev->SetNext(tmp->Next());  
  
            // Delete the current node  
            delete tmp;  
        }  
    }  
}
```

```
}

int main()
{
    List list;
    list.Append(100);
    list.Print();
    list.Append(200);
    list.Print();
    list.Append(300);
    list.Print();
    list.Append(400);
    list.Print();
    list.Append(500);
    list.Print();
    list.Delete(400);
    list.Print();
    list.Delete(300);
    list.Print();
    list.Delete(200);
    list.Print();
    list.Delete(500);
    list.Print();
    list.Delete(100);
    list.Print();
}
```

**Hasil Output**

```
F:\Algo\ok\bin\Debug\ok.exe
100 --> EMPTY
100 --> 200 --> NULL
100 --> 200 --> NULL
100 --> 200 --> 400 --> NULL
100 --> 200 --> 400 --> 500 --> NULL
100 --> 200 --> 500 --> NULL
100 --> 200 --> 500 --> NULL
100 --> 500 --> NULL
100 --> EMPTY
NULL
Process returned 0 (0x0)   execution time : 0.280 s
Press any key to continue.
```

2. Ubah program sehingga menjadi seperti hasil output dibawah ini  
(Gunakan Listing program 1)

```
F:\Algo\ok\bin\Debug\ok.exe
100 --> NULL
100 --> 200 --> NULL
100 --> 200 --> 300 --> NULL
100 --> 200 --> 300 --> 400 --> NULL
100 --> 200 --> 300 --> 400 --> 500 --> NULL
100 --> 200 --> 300 --> 500 --> NULL
100 --> 200 --> 500 --> NULL
100 --> 500 --> NULL
100 --> NULL
EMPTY
Process returned 0 (0x0)   execution time : 3.283 s
Press any key to continue.
```

- 3.
4. Ubah program menjadi bahasa java

## MODUL 5

### LINKED LIST, STACK DAN QUEUE

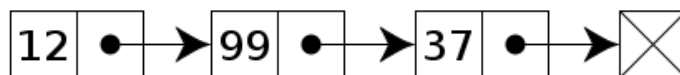
#### 1.1 Tujuan

- Mengetahui penggunaan metode LINKED LIST
- Mengetahui penggunaan metode STACK
- Mengetahui penggunaan metode QUEUE

#### 1.2 Dasar Teori

##### METODE LIST

List atau senarai adalah sebuah pemikiran/ konsep struktur data yang sangat dasar pada pemrograman agar lebih fleksibel, dimana setiap elemen akan ditambahkan saat dibutuhkan. struktur data ini digunakan untuk menyimpan sejumlah objek data biasanya secara terurut sehingga memungkinkan penambahan, pengurangan, dan pencarian atas elemen data yang tersimpan dalam senarai dilakukan secara lebih efektif.



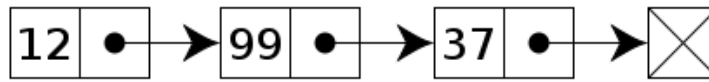
*Sebuah senarai berantai dengan tiap-tiap node yang terdiri atas dua elemen, data integer, dan elemen rujukan ke node berikutnya*

Jenis – jenis Senarai berantai ada 3:

##### 1. Senarai Tunggal

Bila struktur data sebuah node hanya memiliki satu tautan atas node berikutnya dalam sebuah senarai, maka senarai tersebut dinamakan sebagai senarai tunggal.

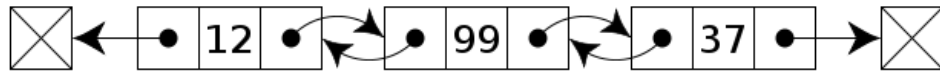




*Senarai tunggal dengan tiap-tiap node yang terdiri atas dua elemen, data integer, dan elemen rujukan ke node berikutnya*

## 2. Senarai Ganda

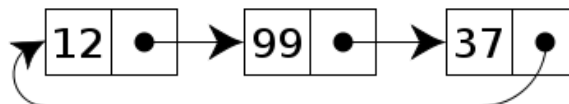
Berbeda halnya dengan senarai tunggal, pada senarai ganda, struktur data atas tiap-tiap node memiliki rujukan pada node sebelum dan berikutnya. Sebagian algoritma membutuhkan taut ganda, contohnya sorting dan reverse traversing.



*Senarai ganda dengan tiap-tiap node yang terdiri atas tiga elemen, data integer, dan dua elemen rujukan ke node sebelum serta berikutnya*

## 3. Senarai Sirkular

Pada dua jenis senarai sebelumnya, node terakhir dalam senarai tersebut merujuk pada **null** yang artinya akhir dari sebuah senarai, begitu pula **null** sebagai rujukan node sebelumnya pada node pertama bila senarai yang dimaksudkan adalah senarai ganda. Pada senarai sirkular, informasi rujukan pada node terakhir akan merujuk pada node pertama, dan rujukan pada node pertama akan merujuk pada node terakhir bila yang digunakan sebagai dasar implementasi adalah senarai ganda.



*Senarai sirkular dengan menggunakan model implementasi senarai tunggal. Node terakhir menyimpan rujukan pada node pertama*

## METODE STACK

sebuah koleksi objek yang menggunakan prinsip LIFO (Last In First Out), yaitu data yang terakhir kali dimasukkan akan pertama kali keluar dari tumpukan

tersebut. Tumpukan dapat diimplementasikan sebagai representasi berkait atau kontigu (dengan tabel fix). Ciri tumpukan:

- TOP merupakan sebutan untuk elemen paling atas dari suatu stack
- Elemen TOP merupakan elemen yang paling akhir ditambahkan
- Elemen TOP diketahui
- penambahan dan penghapusan elemen selalu dilakukan di TOP
- LIFO

Pemanfaatan tumpukan:

- Perhitungan ekspresi aritmatika (posfix)
- algoritma backtracking (runut balik)
- algoritma rekursif

## **METODE QUEUE**

Queue/antrian adalah ordered list dengan penyisipan di satu ujung, sedang penghapusan di ujung lain. Ujung penyisipan biasa disebut rear/tail, sedang ujung penghapusan disebut front/head. Fenomena yang muncul adalah elemen yang lebih dulu disisipkan akan juga lebih dulu diambil. Queue berdisiplin FIFO (First In, First Out). Queue merupakan kasus khusus ordered list. Dengan karakteristik terbatas itu maka kita dapat melakukan optimasi representasi ADT Queue untuk memperoleh kerja paling optimal.

### **1.3 Latihan**

#### **a. Latihan 1**

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

```

using namespace std;
typedef struct{
char nim[10];
char nama[50];
float nilai;
}nilaiMatKul;

typedef struct{
int first;
int last;
nilaiMatKul data[10];
}queue;

void createEmpty(queue*Q){
(*Q).first = -1;
(*Q).last = -1;
}

int isEmpty(queue Q){
    int hasil = 0;
    if(Q.first == -1){
        hasil=1;
    }
    return hasil;
}

int isFull(queue Q){
int hasil = 0;
if(Q.last==9){

```

```

        hasil=1;
    }
    return hasil;
}

void add(char nim[], char nama[],float nilai,
queue *Q){
    if(isEmpty(*Q) == 1 ){
        (*Q).first=0;
        (*Q).last=0;
        strcpy((*Q).data[0].nim,nim);
        strcpy((*Q).data[0].nama,nama);
        (*Q).data[0].nilai=nilai;

    }
    else{
        if(isFull(*Q) !=1){
            (*Q).last=(*Q).last+1;
            strcpy((*Q).data[(*Q).last].nim,nim);
            strcpy((*Q).data[(*Q).last].nama,nama);
            (*Q).data[(*Q).last].nilai = nilai;
        }
        else{
            printf("queue penuh\n");
        }
    }
}

void del(queue *Q){
    if((*Q).last ==0){

```

```

        (*Q).first ==-1;
        (*Q).last ==-1;
    }
else{
    int i;
    for(i=( *Q ).first+1 ; i<= (*Q ).last ; i++){
        strcpy(( *Q ).data[i-
1].nim, (*Q ).data[i].nim);
        strcpy(( *Q ).data[i-
1].nama, (*Q ).data[i].nama);
        (*Q ).data[i-1].nilai=(*Q ).data[i].nilai;
    }
    (*Q ).last=(*Q ).last-1;
}
}

void printfQueue(queue Q){
if (Q.first != -1){
    printf("-----isi queue-----\n");
    int i;
    for(i=Q.last; i>=Q.first; i--){
        printf("elemen ke : %d\n", i);
        printf("nim : %s\n", Q.data[i].nim);
        printf("nama: %s\n", Q.data[i].nama);
        printf("nilai: %f\n", Q.data[i].nilai);
    }
    printf("-----\n");
}
else{

```

```

        printf("queue kosong\n");
    }
}

int main(){
    queue Q;
    createEmpty(&Q);
    printfQueue(Q);

    printf("-----\n");
    add("13507701", "Rangga", "64.67", &Q);
    add("13507702", "Anggy", "75.11", &Q);
    add("13507703", "Intan", "84.63", &Q);
    printfQueue(Q);

    printf("-----\n");
    del(&Q);
    del(&Q);
    printfQueue(Q);
    printf("-----\n");
    return 0;
}

```

## **b. Latihan 2**

```

#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

struct DAT
{
    int id;

```

```

        char fname[20];
        char mname[20];
        char lname[20];
        char address[80];
        double salary;
        char tele_no[15];
};

struct NODE
{
    DAT data;
    NODE *N;
    NODE*P;
    NODE(const int i , const char *f, const char *m, const char
    *l, const char *ad, const double s, const char *tel)
    {
        data.id = i;
        strcpy(data.fname,f);
        strcpy(data.mname,m);
        strcpy(data.lname,l);
        strcpy(data.address,ad);
        data.salary = s;
        strcpy(data.tele_no,tel);
        N = NULL;
        P = NULL;
    }
};

class StackLinkedList
{
private:
    NODE *front;
public:
    StackLinkedList(){front = NULL;}
    ~StackLinkedList(){destroyList();}
    void push(NODE *);
    NODE* pop();
    void destroyList();
};

void StackLinkedList::push(NODE *n)
{
    if(front == NULL)
    {
        front = n;
    }
    else
    {
        front->P = n;
        n->N = front;
        front = n;
    }
}

```

```

}

NODE* StackLinkedList::pop()
{
    NODE *temp;
    if( front == NULL )//no nodes
        return NULL;
    else if(front->N == NULL)//there is only one node
    {
        NODE * temp2 = front;
        temp = temp2;
        front = NULL;
        delete temp2;
        return temp;
    }
    else//there are more than one node
    {
        NODE * temp2 = front;
        temp = temp2;
        front = front->N;
        front->P = NULL;
        delete temp2;
        return temp;
    }
}

void StackLinkedList::destroyList()
{
    while(front != NULL)
    {
        NODE *temp = front;
        front = front->N;
        delete temp;
    }
}

void disp(NODE *N)
{
    if( N == NULL )
    {
        cout << "\nStack is Empty!!!" << endl;
    }
    else
    {
        cout << "\nId No.      : " << N->data.id << " ";
        cout << "\nFull Name   : " << N->data.fname << " ";
        cout << N->data.mname << " ";
        cout << N->data.lname << endl;
        cout << "Address      : " << N->data.address << endl;
        cout << "Salary       : " << setprecision(15) << N->data.salary << endl;
    }
}

```



```

        cout << "Tele_no      : " << N->data.tele_no<< endl <<
endl;
    }
}

int main()
{
    StackLinkedList *Stack = new StackLinkedList();

    NODE No1(101,"Anggy","Eka","P","Semarang
120",7851243.9475,"07502334121");
    NODE No2(102,"Wa Ode","Kanartia","Ningsi","Timor leste
121",5681125.9457,"07507534127");
    NODE No3(103,"Fatimah","Nurul","Intan","Irian Jaya
123",2344003.48345,"075078654129");

    Stack->push(&No1);
    Stack->push(&No2);
    Stack->push(&No3);

    disp(Stack->pop());

    disp(Stack->pop());

    disp(Stack->pop());

    disp(Stack->pop());

    delete Stack;
    return 0;
}

```

## 1.4 Tugas

1. Benarkan listing program sehingga dapat dicompile !

```

#include <iostream>
using namespace std
class node {
    int data;
    Node* next;

```

```

public:
    Node() {};
    void SetData(int aData) { data = aData; };
    void SetNext(Node* aNext) { next = aNext; };
    int Data() { return data; };
    Node* Next() { return next; }
};

```

```

class List {
    Node *head;
public:
    List() { head = NULL; };
    void Print();
    void Append(int data);
    void Delete(int data);
};

```

```

void List::Print() {
    Node *tmp = head;
    if ( tmp == NULL ) {
        cout << "NULL" << endl;
        return;
    }
    if ( tmp->Next() == NULL ) {
        cout << tmp->Data();
        cout << " --> ";
    }
}

```

```

        cout << "EMPTY" << endl;
    }
    else {
        do {
            cout << tmp->Data();
            cout << " --> ";
            tmp = tmp->Next();
        }
        while ( tmp != NULL );

        cout << "NULL" << endl;
    }
}

void List::Append(int data) {
    if(data != 300) {
        Node* newNode = new Node();
        newNode->SetData(data);
        newNode->SetNext(NULL);

        Node *tmp = head;
        if ( tmp != NULL ) {
            while ( tmp->Next() != NULL ) {
                tmp = tmp->Next();
            }
            tmp->SetNext(newNode);
        }
        else {
            head = newNode;
        }
    }
}

```

```
    }  
    }  
}
```

```
void List::Delete(int data) {  
  
    if(data != 300) {  
        Node *tmp = head;  
        if ( tmp == NULL )  
            return;  
        if ( tmp->Next() == NULL ) {  
            delete tmp;  
            head = NULL;  
        }  
        else {  
            Node *prev;  
            do {  
                if ( tmp->Data() == data ) break;  
                prev = tmp;  
                tmp = tmp->Next();  
            } while ( tmp != NULL );  
  
            // Adjust the pointers  
            prev->SetNext(tmp->Next());  
  
            // Delete the current node  
            delete tmp;  
        }  
    }  
}
```

```
}

int main()
{
    List list;
    list.Append(100);
    list.Print();
    list.Append(200);
    list.Print();
    list.Append(300);
    list.Print();
    list.Append(400);
    list.Print();
    list.Append(500);
    list.Print();
    list.Delete(400);
    list.Print();
    list.Delete(300);
    list.Print();
    list.Delete(200);
    list.Print();
    list.Delete(500);
    list.Print();
    list.Delete(100);
    list.Print();
}
```

**Hasil Output**

```
F:\Algo\ok\bin\Debug\ok.exe
100 --> EMPTY
100 --> 200 --> NULL
100 --> 200 --> NULL
100 --> 200 --> 400 --> NULL
100 --> 200 --> 400 --> 500 --> NULL
100 --> 200 --> 500 --> NULL
100 --> 200 --> 500 --> NULL
100 --> 500 --> NULL
100 --> EMPTY
NULL
Process returned 0 (0x0)   execution time : 0.280 s
Press any key to continue.
```

- Ubah program sehingga menjadi seperti hasil output dibawah ini  
(Gunakan Listing program 1)

```
F:\Algo\ok\bin\Debug\ok.exe
100 --> NULL
100 --> 200 --> NULL
100 --> 200 --> 300 --> NULL
100 --> 200 --> 300 --> 400 --> NULL
100 --> 200 --> 300 --> 400 --> 500 --> NULL
100 --> 200 --> 300 --> 500 --> NULL
100 --> 200 --> 500 --> NULL
100 --> 500 --> NULL
100 --> NULL
EMPTY
Process returned 0 (0x0)   execution time : 3.283 s
Press any key to continue.
```

- 
- 
- 
- Ubah program menjadi bahasa java