KONSTANTA DAN VARIABEL

KONSTANTA

Bilangan numerik atau suatu kalimat string dapat dijadikan sebagai konstanta.

Manfaat Konstanta

Dengan menggunakan konstanta dapat memberikan nama yang mudah dipahami, misal akan lebih mudah menulis pi, daripada harus menuliskan 3,1415926536 berkali-kali diprogram.

Deklarasi Konstanta

Konstanta dideklarasikan pada awal program sebelum blok **begin-end** program utama dituliskan. Untuk mendeklarasikan konstanta harus diawali dengan kata baku **const**.

Bentuk Umum :

```
Const
   NamaKonstanta1 = NilaiKonstanta1
   NamaKonstanta2 = NilaiKontansta2
   .........
   NamaKonstantaN = NilaiKonstantaN
```

Contoh :

```
Const
   Pi              = 3,14;
   Radius          = 25,6;
   LuasLingkaran   = pi*radius;
```

Konstanta Bertipe

Konstanta bertipe adalah suatu konstanta yang dideklarasikan dengan tipe tertentu.

Bentuk Umum :

```
Const
   NamaKonstanta1  : Tipe1  = NilaiKonstanta1;
   NamaKonstanta2  : Tipe2  = Nilaikonstanta2;
   ................
   NamaKonstantaN  : TipeN  = NilaiKonstantaN;
```

Contoh :

```
Const
   JumMhs          : integer = 15000;
   CheckPosisi     : boolean = True;
   BanyakData      : byte = 250;
```

VARIABEL

Variabel adalah suatu lokasi di memori yang disiapkan oleh programer dan diberi nama yang khas untuk menampung suatu nilai.

Bentuk Umum :

```
Var
   NamaVariabel11,
   NamaVariabel12,
   ...
   NamaVariabel1N   : TipeData1;
   NamaVaraibel21,
   NamaVariabel22,
   ...
   NamaVariabelNN   : TipeDataN;
```

Contoh :

```
Var
   Nilai1,
   Nilai2,
   Nilai3    : byte;
   JmlData : integer;
```

Tipe Data Sederhana

Pada saat mendeklarasikan variabel secara otomatis harus mendeklarasikan tipe data yang dapat ditampung oleh variabel tersebut.

Macam – macam tipe data sederhana :

- Integer

  Tipe data integer adalah tipe data yang nilainya merupakan bilangan bulat.

- Boolean

  Tipe data boolean biasa digunkan untuk mempersentasikan logika. Tipe data boolean hanya bernilai True atau False.

- Real

  Tipe data real biasa digunakan untuk mempersentasikkan nilai pecahan.

- Karakter

  Tipe data karakter hanya dapat menampung satu karakter saja.

- String

  Tipe data string merupakan gabungan dari karakter sebanyak 256(default).

UNIT DAN INPUT/OUTPUT

UNIT

Unit adalah kumpulan procedure dan function yang siap dipakai dengan hanya mendeklarasikan nama unitnya pada awal program.

Unit dikelompokkan menjadi beberapa bagian :

- Unit System

  Unit system ini berisi procedure dan function standar yang sudah siap pakai. untuk menggunakan preintah yang tergabung dalam unit system ini tidak perlu mendeklarasikan dengan menggunkan kata uses karena secara otomatis telah didkelarasikan oleh pascal.

- Unit CRT

  Unit CRT ini berisi kumpulan prcedure dan function yang berguna untuk memanipulasi tampilan pada laya monitor (dlam mode text) dan yang berhubungan dengan penekanan tombol keyboard. Untuk memakai perintah ini harus mendeklarasikan dengan **uses crt**.

- Unit DOS

  Unit DOS berisi kumpulan procedure dan function yang berguna untuk melakukan proses yang berhubungan dengan fungsi DOS. Untuk medeklarasikannya harus di awali dengan **uses** dan diikuti oleh **DOS**

- Unit String

  Unit string berisi kumpulan procedure dan function yang berguna untuk memanipulasi hal-hal yang mengggunakan tipe data string. Untuk memakainya juga perlu dideklarasikan dengan **uses string**.

- Unit Graph

  Unit graph berisi kumpulan procedure dan function yang berguna untk membuat grafik (gambar). Untuk memakai ini harus mendeklarasikan dengan **uses graph**.

Bentuk umum :

uses <NamaUnit>;

Contoh :

uses Crt, DOS, Graph;

INPUT DAN OUTPUT

1. Write dan Writeln

   Untuk menampilkan hasil ke layar dapat menggunakan printah write dan writeln. Write dan writeln mempunyai sedikit perbedaan. Perbedaannya adalah write akan menampilkan hasil ke layar tanpa disertai ganti baris sedangkan writeln akan menampilkan hasil ke layar dengan disertai pergantian baris.

   Bentuk umum :

   Write (variabel | string | numerik);
   Writeln (variabel | string | numerik);

   Contoh :

   Wriite ('ZAY');
   Write (123456);
   Writeln ('*****');

   Write (A);
   Writeln (JumlahMhs);

2. Read dan Readln

   Untuk meminta masukan dari keyboard, dapat memanfaatkan procedure read dan readln. Perbedaanya jika readln, setelah data dimasukkan dan harus menekan tombol enter akan disertai dengan pergantian baris sedangkan read tidak.

   Bentuk umum :

   Read (Variabel);
   Readln (vaiabel);

   Contoh :

   Read (NamaSiswa);
   Readln (NPM);

Contoh Soal :

```
Var   bil  : byte;
      kata : string (25);
      kar : char;
Begin
      Write ('masukan sebuah bilangan :'); Readln (bil);
      Write ('masukan sebuah kata :'); Readln (kata);
      Write ('masukan suatu karakter :'); Readln(kar);
      Writeln ('bilangan anda adalah : ',bil);
      Writeln ('kata anda adalah : ',kata);
      Writeln ('karakter anda adalah : ',kar);
End.
```

RECORD

Record adalah kumpulan elemen-elemen data yang digabungkan menjadi satu kesatuan. Masing masing elemen data tersebut dikenal dengan sebutan field. Field memiliki tipe data yang sama ataupun berbeda. Walaupun field tersebut berada dalam satu kesatuan record namun field-field tersebut tetap dapat diakses secara individual.

Deklarasi Record

Bentuk Umum :

```
type
   <NamaRecord> = record
                    <Data Field1>:<type1>;
                    <Data Field2>:<type2>;
                    .....
                    <Data FieldN>:<typeN>;
                  end.
Var
   <NamaVariabel>:<Nama Record>;
```

Contoh :

```
type
   Mahasiswa =  record
                    NIM     : string [10];
                    Nama    : string [20];
                    Alamat  : string [30];
                    IPK     : real;
                End.

var
   Mhs : Mahasiswa;
```

Pemakaian Record

Menggunakan record harus ditulis nama record beserta dengan fieldnya yang dipisahkan dengan tanda titik (.).

Contoh :

```
Write (Mhs.NIM);
```

Contoh Soal :

```
Program data mhs;
Uses crt;
Type
        Mahasiswa = record
Nama  : string [20];
NIM   : string [30];
Alamat: string [50];
IPK   : real;
        End.

Begin
    Clrscr;
    Write ('Nama  : '); Readln(Mhs.Nama);
    Write ('NIM   : '); Readln(Mhs.NIM);
    Write ('Alamat : '); Readln(Mhs.Alamat);
    Write ('IPK   : '); Readln(Mhs.IPK);
    Writeln;
    Writeln ('Nama Anda    : ',Mhs.Nama);
    Writeln ('NIM Anda     : ',Mhs.NIM);
    Writeln ('Alamat Anda  : ',Mhs.Alamat);
    Writeln ('IPK Anda     : ',Mhs.IPK:2:2);
End.
```

ARRAY

Array adalah suatu tipe data tersetruktur yang terdapat dalam memory yang terdiri dari sejumlah elemen (tempat) yang mempunyai tipe data yang sama dan merupakan gabungan dari beberapa variabel sejenis.

Elemen – elemen dari array tersusun secara sequential dalam memory komputer. Array dapat berupa satu dimensi, dua dimensi tiga dimensi ataupun banyak dimensi.

Array Satu Dimensi

Kumpulan dari elemen – elemen yang identik yang tersusun dalam satu baris, elemen tersebut memiliki tipe data yang sama, tetapi isi dari elemen tersebut boleh berbeda.

Konsep Array Satu Dimensi

(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
17	21	33	1	48	0	2	16	72	9

Bentuk Umum

Type

<NamaArray> = array[IndexArray] of TipeData;

Contoh

Type

Gaji = array[1..10] of longint;
Logika = array[boolean] of integer;

Pendeklarasian di awali dengan kata baku **type** diikuti dengan nama array dan tanda samadengan (=), lalu kata baku **array** beserta range index dan di akhiri dengan kata baku **of** beserta tipe datanya.

Selain itu konstanta juga dapat dipakai dalam index array. Untuk menggunakan konstanta perlu di awali dengan kata baku const. Perhatikan contoh :

```
Const
    Min = 1;
    Max = 10;
Type
    Arr = array [Min..Max] of byte;
Var
    Point = arr;
```

Contoh soal :

```
Uses Crt;
Type
    Kalimat = Array[1..3] of string;
Var
    i : byte;
    Kal : kalimat;
Begin
  Clrscr;
  For i:= 1 to 3 do
    Begin
      Write (‘masukan kata-‘,i,’:  ‘); Readln(kal[i]);
    End.
End.
```

Array Dua Dimensi

Array dua dimensi sering digambarkan dalam matriks merupakan perluasan dari array dimensi satu. Jika pada array satu dimensi hanya terdiri dari sebuah baris dengan beberapa kolom maka pada array dua dimensi terdiri dari baris dan beberapa kolom yang betipe sama.

Konsep Array Dua Dimensi

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
[1]	12	34	45	43	23	7	6	5	67
[2]	34	45	5	16	12	90	45	34	64
[3]	45	34	76	23	45	67	98	76	65

Bentuk Umum

```
Type
  <NamaArray> = array[IndexArray1,IndexArray2] of TipeData
```

Contoh 1 :

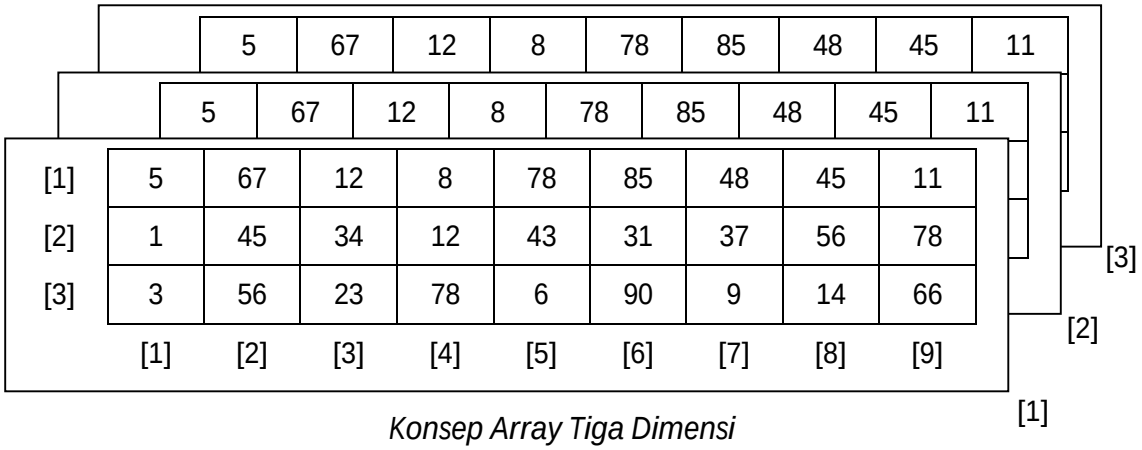
```
Type
  Matriks = array [1..2,1..3] of byte;
  Logika = array [1..5,boolean] of integer;

Type
  Baris = 1..2;
  Kolom = 1..3;
  Ordo = array[Baris,kolom] of byte;
Var
  Matriks : Ordo;
```

Contoh 2 :

```
Type
    Kegiatan  = (main, belajar, nonton, berenang);
    Hari      = (senin, selasa, rabu, kamis, jum'at, sabtu);
    Aktivitas = array[Hari,Kegiatan] of byte;
Var
    A: Aktivitas;
```

Array Tiga Dimensi

Array tiga dimensi dapat digambarkan sebagai suatu benda runag seperti berikut di bawah ini :



Konsep Array Tiga Dimensi

Bentuk Umum

```
Type
    <NamaArray> = array[IndexArray,IndexArray2,IndexArray3] of TipeData;
```

Contoh :

```
Type
    Kalender  = array [tanggal,bulan,tahun] of byte;
    Logika    = array [1..10,boolean,2..15] of integer;
```

Array Banyak Dimensi

Array banyak dimensi pada dasarnya sama dengan array sebelumnya kecuali pada jumlah dimensinya saja.

Bentuk Umum

```
Type
    <NamaArray> = array<IndexArray1,IndexArray2,...,IndexArrayN) of TipeData;
```

Contoh

```
Type
    Waktu = array[tahun, bulan, tanggal, jam] of integer;
```

Pemakaian Array (Array Satu Dimensi)

Misalkan terdapat array satu dimensi sebagai berkut :

Var Nilai: **Array** [0..9] **of** byte;

Index :	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)

Tiap tempat dapat menampung data yang bertipe sama yaitu byte [0..255]. Jika ingin mengisi tempat ke-5 dengan nilai 75, maka dituliskan sebagai berikut :

Nilai[5]=75;

Sehingga menjadi :

Index :	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
						75				

Sedangkan jika ingin mengambil kembali nilai elemen pada index ke-5 maka dilakukan sbb:

Temp:=Nilai[5]

SORT

Sort adalah proses pengurutan data yang sbelumnya disusun secara acak sehingga menjadi tersusun secara teratur menurut suatu aturan tertentu.

Ada 2 jenis pengurutan :

- Ascending (naik)
- Descending (turun).

Contoh :

Data acak	:	5	6	8	1	3	25	10
Terurut Acending	:	1	3	5	6	8	10	25
Terurut Descending	:	25	10	8	6	5	3	1

Untuk melakukan pengurutan ada beberapa macam cara, di antaranya :

1. Bubble/Exchange Sort
2. Selection ort
3. Insertion Sort
4. Quick Sort

BUBBLE/EXCHANGE SORT

Membandingkan elemen yang sekarang dengan elemen yang berikutnya, jika elemen sekarang > elemen berikutnya, maka tukar

Proses :

Langkah 1 :

22	10	15	3	8	2
22	10	15	3	2	8
22	10	15	2	3	8
22	10	2	15	3	8
22	2	10	15	3	8
2	22	10	15	3	8

Pengecekan dapat dimulai dari data paling awal atau paling akhir. Pada contoh di samping ini pengecekan dimulai dari data yang paling akhir dibandingkan dengan data didepannya, jika ternyata lebih kecil maka tukar. Dan pengecekan yang sama dilakukan terhadap data yang selanjutnya sampai dengan data yang paling awal.

Langkah 2 :

2 22 10 15 3 8

2 22 10 15 3 8

2 22 10 3 15 8

2 22 3 10 15 8

2 3 22 10 15 8

Kembali data paling akhir dibandingkan dengan data didepannya jika ternyata lebih kecil maka tukar, tapi kali ini pengecekan tidak dilakukan sampai data paling awal yaitu 2 karena data tersebut merupakan data terkecil.

Langkah 3 :

2 3 22 10 15 8

2 3 22 10 8 15

2 3 22 8 10 15

2 3 8 22 10 15

Langkah 4 :

2 3 8 22 10 15

2 3 8 22 10 15

2 3 8 10 22 15

Langkah 5 :

2 3 8 10 22 15

2 3 8 10 15 22

Langkah 6 :

2 3 8 10 15 22

Proses di atas adalah pengurutan data dengan metode bubble ascending. Untuk descending adalah kebalikan dari proses di atas. Berikut listing program Procedure TukarData dan Procedure Bubble Sort

```
Procedure TukarData (Var a,b: word);
Var c : word;
   Begin
     c:=a;
     a:=b;
     b:=c;
   End.

Procedure Asc_Bubble(var data:array; jmldata:integer);
Var
   i,j:integer;
   Bagin
     for i:= 2 to jmldata do
     for j:= jmldata downto i do
       if data[j] < data [j-1] then
```

```
        TukarData (data[j], data[j-1]);
End.
```

if data[j] > data [j-1] untuk pengurutan secara descending.

SELECTION SORT

Membandingkan elemen yang sekarang dengan elemen yang berikutnya sampai dengan elemen yang terakhir. Jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka dicatat posisinya dan kemudian ditukar. Dan begitu seterusnya. **Proses :**

Langkah 1 :

(1) (2) (3) (4) (5) (6)

22 10 15 3 8 2

Pembanding	Posisi
22 > 10	2
10 <15	2
10 > 3	4
3 < 8	4
3 >2	6

Posisi data ke-1 (22) = 6,

Tukar data ke-1 dengan data ke-6

2 10 15 3 8 22

Langkah 2 :

(1) (2) (3) (4) (5) (6)

2 10 15 3 8 22

Pembanding	Posisi
10 < 15	2
10 > 3	4
3 < 8	4
3 < 22	6

Posisi data ke-2 (10) = 4,

Tukar data ke-2 dengan data ke-4

2 3 15 10 8 22

Langkah 3 :

(1) (2) (3) (4) (5) (6)

2 3 15 10 8 22

Pembanding	Posisi
15 > 10	4
10 > 8	5
8 < 22	5

Posisi data ke-3 (15) = 5,

Tukar data ke-2 dengan data ke-5

2 3 15 10 8 22

Langkah 4 :

(1) (2) (3) (4) (5) (6)

2 3 8 10 15 22

Pembanding	Posisi
10 < 15	4
10 < 22	4

Posisi data ke-4 tetap

Pada posisinya = 4 (tidak berubah)

2 3 8 10 15 22

Langkah 5 :

(1) (2) (3) (4) (5) (6)

2 3 8 10 15 22

Pembanding	Posisi
15 < 22	5

Posisi data ke-5 tetap

Pada posisinya = 5 (tidak berubah)

Terurut : 2 3 8 10 15 22

(Proses pengurutan diatas adalah dengan metode Selection Ascending. Untuk Descending hanyalah kebalikan dari proses di atas. Berikut contoh listing Program Procedure Selection Sort secara ascending.

```
Procedure Asc_Selection;
Var
   min,pos: byte;
   Begin
      For i:= 1 to max-1 do
         Begin
            Pos:=1;
         For j:= i+1 to max do
            If data[j] < data[pos] then pos:=j;
            If i<>pos then TukarData(data[i],data[pos]);
         End;
   End.
```

Untuk pengurutan secara descending, hanya perlu mangganti baris ke-8 sebagai berikut :

```
            If data[pos] < data[j] then pos:=j;
```

INSERTION SORT

Pengurutan dilakukan dengan cara membandingkan data ke-i (dimana i dimulai dari data ke-2 sampai dengan data terakhir) dengan data berikutnya. Jika ditemukan data yang lebih kecil maka data tersebut disisipkan kedepan sesuai dengan posisi yang seharusnya.

Proses :

Langkah 1 :

(1)	(2)	(3)	(4)	(5)	(6)
22	10	15	3	8	2

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

Temp menempati posisi ke-1

10	22	15	3	8	2

Langkah 2 :

(1)	(2)	(3)	(4)	(5)	(6)
10	22	15	3	8	2

Temp	Cek	Geser
15	Temp < 22	Data ke-2 -> Posisi 3
	Temp > 22	-

Temp menempati posisi ke-2

10	15	22	3	8	2

Langkah 3 :

(1)	(2)	(3)	(4)	(5)	(6)
10	15	22	3	8	2

Temp	Cek	Geser
3	Temp < 22	Data ke-3 -> Posisi 4
	Temp < 15	Data ke-3 -> Posisi 4

Temp menempati posisi ke-2

10	22	15	3	8	2

Langkah 4 :

(1)	(2)	(3)	(4)	(5)	(6)
3	10	15	22	8	2

Temp	Cek	Geser
8	Temp < 22	Data ke-4 -> Posisi 5
	Temp < 15	Data ke-3 -> Posisi 4
	Temp < 10	Data ke-2 -> Posisi 2
	Temp > 3	

Temp menempati posisi ke-2

3	8	10	15	22	2

Langkah 5 :

(1)	(2)	(3)	(4)	(5)	(6)
3	8	10	15	22	2

Temp	Cek	Geser
2	Temp < 22	Data ke-5 -> Posisi 6
	Temp < 15	Data ke-4 -> Posisi 5
	Temp < 10	Data ke-3 -> Posisi 4
	Temp < 8	Data ke-2 -> Posisi 3
	Temp < 3	Data ke-1 -> Posisi 2

Temp menempati posisi ke-1

Terurut :

2	3	8	10	15	22

Prosedure Insretion Sort Ascending

```
Procedure Asc_Insert;
Var
   i,j,temp : byte;
   begin
   For i:= 2 to max do
      Begin
         temp:=data[i];
         j:=i-1;
         While(data[j]>temp) and (j>0) do
            Begin
               data[j+1]:=data[j];
               dec[j];
            end;
         data[j+1]:=temp;
      end;
   End.
```

Untuk pengurutan secara descending hanya mengganti baris ke-8 dengan baris berikut :

```
While(data[j]>temp) and (j>0) do
```

QUICK SORT

Membandingkan suatu elemen (pivot) dengan elemen yang lain dan menyusunnya sedemikian rupa sehingga elemen-elemen lain yang lebih kecil daripada pivot tersebut disebelah kirinya dan elemen – elemen yang lain yang lebih besar daripada pivot tersebut disebelah kanannya. Sehingga dengan demikian telah terbentuk sublist yang terletak disebelah kiri dan kanan dari pivot. Lalu pada sublist kiri dan sublist kanan kita anggap sebagai list baru dan kita kerjakan proses yang sama seperti sebelumnya.

Proses :

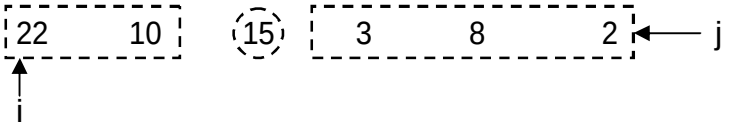
Bilangan yang di dalam kurung merupakan *pivot*

Persegi Panjang yang digambarkan dengan garis terputus-putus menunjukkan sublist

i bergerak dari sudut kiri ke kanan sampai mendapatkan nilai yang >= pivot

j bergerak dari sudut kanan ke kiri sampai menemukan nilai yang < pivot

Langkah 1 :



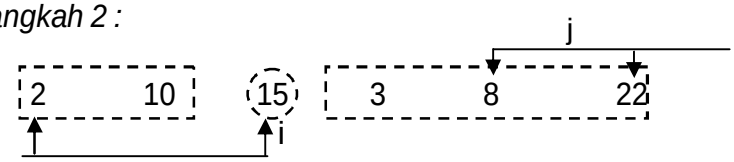
i berhenti pada index ke-1 karena langsung mendapatkan nilai yang > dari pivot

j berhenti pada index ke-6 karena langsung mendapatkan nilai yang < dari pivot.

Karena i < j maka data yang ditunjuk oleh i ditukar dengan data yang ditunjuk oleh j sehingga menjadi

2 10 15 3 8 22

Langkah 2 :



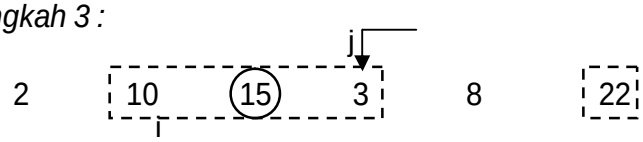
i behenti pada index ke-3 (pivot) karena tidak menemukan bilangan yang > dari pivot

j berhenti pada index ke-5 menunjuk pada nilai yang < dari pivot

karena i < j maka data yang ditunjuk oleh i(pivot) ditukar dengan data yang ditunjuk oleh j sehingga menjadi

2 10 8 3 15 22

Langkah 3 :



Proses yang sama seperti sebelumnya dilakukan terhadap 2 buah sublist yang baru (ditandai dengan persegi panjang dengan garis terputus-putus)

2 3 8 10 15 22

Prosedure Quick Sort dengan nilai paling kiri sebagai pembanding (pivot)

```
Procedure Asc_Quick(L,R : integer);
Var
   i,j : integer;
   begin
   i:=L; j:=R;
      repeat
         repeat inc(i) until data[i] >= data[1];
         repeat dec(j) until data[j] <= data[1];
         if i<j then TukarData(data[i],data[j]);
      until i>j;
         TukarData(data[1],data[j]);
         Asc_Quick(L,j-1);
         Asc_Quick(j+1,R);
      End;
   End;
```

Untuk pengurutan secara descending hanya tinngal mengganti tanda aritmatik pada baris ke-8 dan ke-9 sehingga menjadi sepperti baris berikut :

```
         repeat inc(i) until data[i] <= data[1];
         repeat dec(j) until data[j] >= data[1];
```

Prosedure Quick Sort dengan nilai tengah sebagai pembanding (pivot)

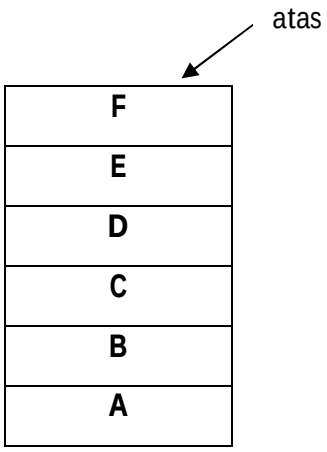
```
Procedure Asc_Quick(L,R:integer);
Var
   mid,i,j : integer;
   Begin
      i:=L;    j:=R;    mid:=data[(L+R) div 2];
      repeat
         while data[i] < mid do inc(i);
         while data[j] > mid do dec(j);
         if i<=j then
            begin
               change(data[i],data[j]);
               inc(i); dec(j);
            end;
      until i>j;
      if L<R then Asc_Quick(L,j);
      if i<R then Asc_Quick(i,R);
   end.
```

Untuk pengurutan secara descending hanya tinggal mengganti baris ke-6 dan ke-7 sehingga menjadi seperti baris berikut :

```
         while data[i] > mid do inc(i);
         while data[j] < mid do dec(j);
```

STACK (TUMPUKAN)

Salah satu bentuk struktur data yang menyerupai tumpukan dimana satu data diletakkan di atas satu data lainnya. Jika dilakukan penambahan (penyisipan) data dan pengambilan (penghapusan) dilakukan lewat ujung yang sama, yang disebut ujung tumpukan. Data yang masuknya palingakhir akan di akses lebih dulu (last in first out).



Pada gambar di atas elemen **F** merupakan elemen teratas. Jika ada data lain yang akan ditambahkan sebagai elemen yang diletakkan setelah **F.**

1. Mengisi Stack (Push)

Penambahan elemen pada stack dapat dilakukan selama stack belum penuh (belum mencapai batas maksimal)

Algoritma :

```
While batas atas < maksimal
Do
Batas atas = batas atas + 1
    Isi[batas atas] = x
End While
    Write ‘stack penuh’
```

\

2. Mengambil isi Stack (Pop)

Operasi ini untuk menhapus elemen yang terletak pada posisi paling atas dari sebuah stack. Operasi pengambilan (penghapusan) ini dapat dilakukan selama stack belum kosong.

Algoritma :

```
While batas atas > 0
Do
    Batas atas = batas atas - 1
End while
    Write 'stack kosong'
```

Impelementasi pada stack antara lain dilakukan pada penulisan numerik. Penulisan ungkapan numerik ada 3 cara yaitu Infiks, Prefiks, dan Postfiks

a. Infiks

Yaitu penulisan ungkapan numerik dilakukan dengan menuliskan operator di antara dua operand. Contoh :

A+B*C

A,B,C adalah Operand dan + * adalah Operator. Pelaksanaan ungkapan tersebut dilakukan berdasarkan level prioritas. Adapun urutan level tertinggi adalah (), ^, / dan *, + dan - .

b. Prefiks

Dilakukan dengan menuliskan operator sebelum operand. Contoh :

A*B^C+(D-E)-F^H

A*B^C+(-DE)-F^H

A*^BC+(-DE)-^FH

*A^BC+-DE-^FH

+*A^BC-DE-^FH

-+*A^BC-DE^FH

c. Postfiks

Dilakukan dengan menuliskan operator setelah operand. Contoh :

A*B^C+(D-E)-F^H

A*B^C+(DE-)-F^H

A*BC^+(DE-)-FH^

ABC^*+(DE-)-FH^

ABC^*(DE-)+-FH^

ABC^*(DE-)+FH^-

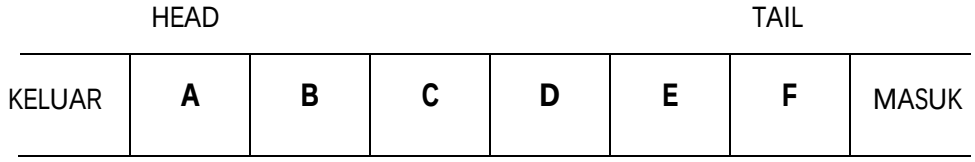
ANTRIAN (QUEUE)

Merupakan suatu kumpulan data yang mana penambahan elemen hanya bisa dilakukan pada ujung yang disebut sisi belakang dan penghapusan atau pengambilan dilakukan lewat ujung yang lainnya disebut dengan sisi depan. Pada antrian prinsio yang digunakan adalah FIFO (First In First Out) yang berarti elemen yang masuk pertama akan keluar pertama.

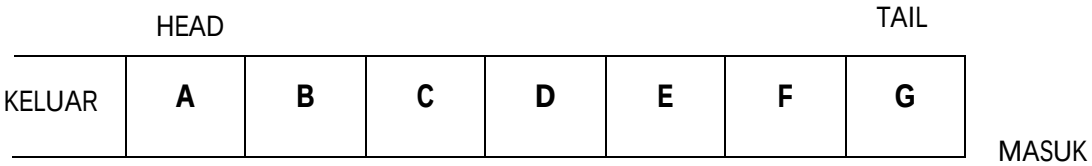
1. Implementasi Antrian

   Pada implementasi menggunakan array, data ditempatkan beurutan dengan jumlah elemen maksimal sesuai jumlah yang didefinisikan.

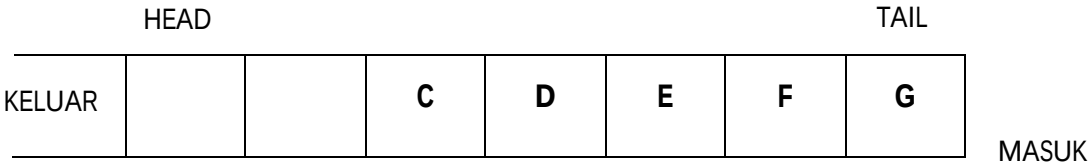
   Contoh :



H = 1, T = 6



H = 1, T = 7



H = 3, T = 7

2. Operasi – operasi pada Antrian

- Create : Suatu operasi yang membuat antrian kosong.

```
Procedure create;
 Begin
    Head :=0; Tail :=0
 End.
```

- Empty : Operasi yang berguna untuk mengecek apakah antrian masih kosong atau sudah berisi.

```
Procedure Empty:
 Begin
    If Tail = 0 then
      Empty :=true;
    Else
       Empty:=false;
End.
```

- Full : Operasi yang berguna untuk mengecek apakah antrian sudah penuh atau masih bisa menampung data.

```
Procedure Full;
 Begin
    If Tail = MaxQueue then
      Full := true;
    Else
      Full := false;
 End.
```

- EnQueue : Operasi ini berguna untuk memasukkan 1 elemen ke dalam antrian.

```
Procedure EnQueue(elemen : byte);
 Begin
    If Empty then
       Begin
          Head := 1;
          Tail   :=1;
          Queue[head] := elemen;
        End.
    Else
    If Not Full then
        Begin
          Inc(Tail);
          Queue(Tail) := elemen;
    End.
End.
```

- DeQueue : Operasi ini berguna untuk mengambil 1 elemen dari antrian.

```
Procedure DeQueue
Var i : byte;
    Begin
        If Not Empty then
            Begin
                For i := Head to Tail-1 do
                  Queue[i] := Queue[i+1];
                Dec(Tail);
            End.
End.
```

- Clear : Operasi ini berguna untuk menghapus semua elemen dalam antrian dengan jalan mengeluarkan smua elemen tersebut atau satu per satu.

```
Procedure clear;
    Begin
        While Not Empty then
            DeQueue;
    End.
```

3. Antrian Sirkular

Apabila terjadi operasi yang dilakukan melebihi jumlah elemen yang didefinisikan (n), maka operasi selanjutnya dapat menggunakan prinsip array sirkular (perputaran).

Contoh :

			I	**H**	**C**	**D**	**B**

Remove (B,D,C,H)

			I				

Insert (J,Q,L,M,N,O)

N	**O**		**I**	**J**	**Q**	**L**	**M**

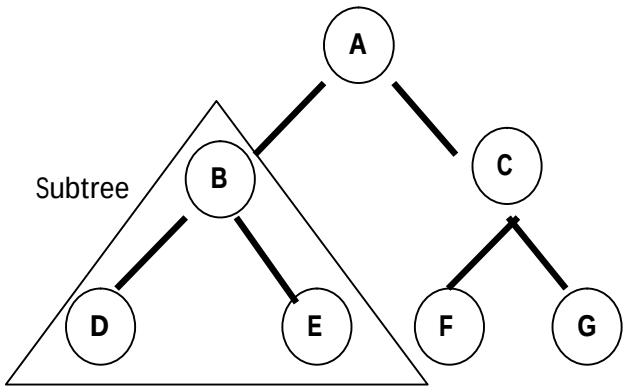
STRUKTUR POHON (TREE)

Merupakan salah satu bentuk struktur data tak linier yang mememiliki ciri-ciri khusus. Struktur ini biasanya digunakan untuk menggambarkan hubungan yang bersifat hierarki(hubungan one to many) antara elemen-elemen. Terdiri dari simpul/node, dimana satu simpul merupakan akar (root) dan simpul-simpul yang lain membentuk suatu subtree

Istilah-istilah umum pada Tree :

1. Predecessor : node yang berada di atas node tertentu
2. Successor : node yang berada di bawah node tertentu
3. Ancestor : seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama
4. Descendant : seluruh node yang terletak sesudah node tertentu dan terletak pada jalur yang sama
5. Parent : predecessor satu level dibawah satu node
6. Child : successor satu level dibawah suatu node
7. Sibling : node-node yang memiliki parent yang sama pada suatu node
8. Subtree : bagian dari tree yang berupa suatu node beserta descendantnya dan memiliki semua karakteristik dari tree tersebut
9. Size : banyanya node dalam suatu tree
10. Height : banyaknya tingkatan/level dalam suatu tree
11. Root : satu-satunya node khusus dalam tree yang tidak punya predecessor
12. Leaf : node-node dalam tree yang tidak punya successor
13. Degree : banyaknya child yang dimilki suatu node

Contoh :



Ancestor (F) = C,A

Descendant (C) = F,G

Parent (D) = B

Child (A) = B,C

Sibling (F) = G

Size = 7

Height = 3

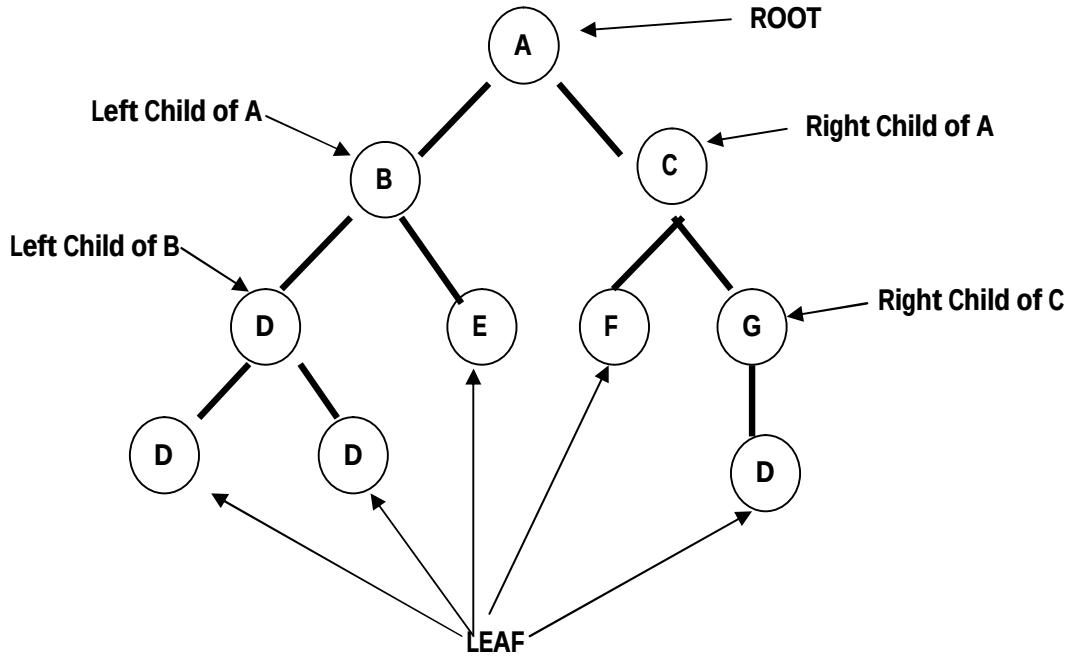
Root = D, E, F, G

Degree (C) = 2

Beberapa jenis Tree yang memiliki sifat khusus

1. Binnary Tree

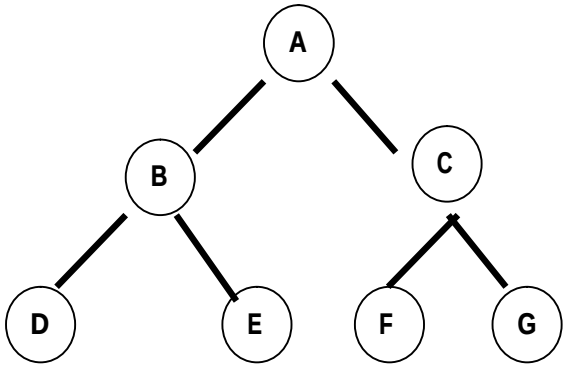
Adalah tree dengan syarat bahwa tiap node hanya boleh memilki maksimal dua subtree dan dua subtree tersebut harus terpisah.



Jenis – jenis Binary Tree :

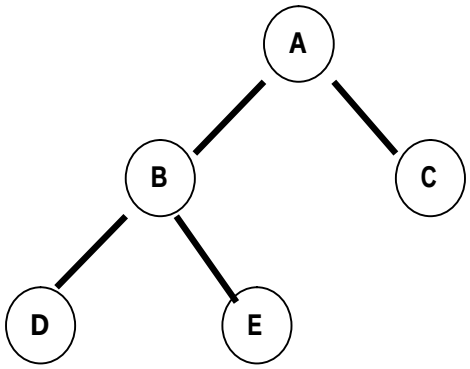
a. Full Binary Tree

Binary tree yang tiap nodenya (kecuali leaf) memilki dua child dan tiap subtree harus mempunyai panjang path yang sama.



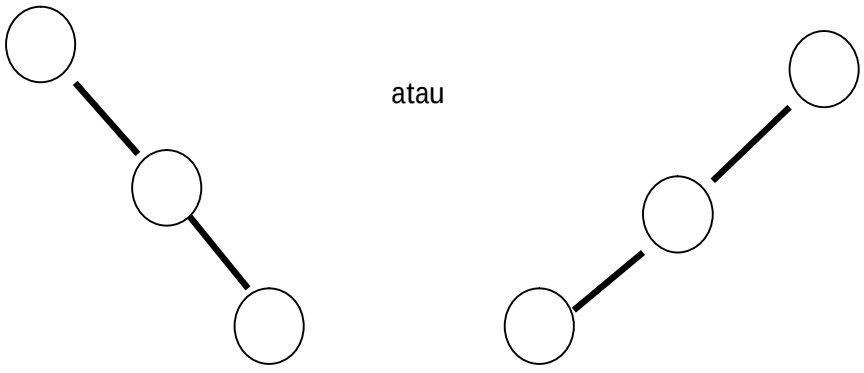
b. Complete Binary Tree

Mirip dengan Full Binary Tree, namun tiap subtree boleh memilki panjang path yang berbeda. Node kecuali leaf memilki 0 atau 2 child



c. Skewed Binary Tree

Bianry Tree yang semua nodenya (kecuali leaf) hanya memliki satu child



Oprerasi – operasi pada Binary Tree :

- Create : membentuk binary tree baru yang masih kosong
- Clear : mengosongkan binary tree yang sudah ada
- Empty : function yang memeriksa binary tree masih kosong
- Insert : memasukkan sebuah node ke dalam tree. Ada tiga pilihan insert : sebagai root, left child atau right. Khusu insert root, tree harus benar-benar kosong.
- Find : mencari root, parent, left child atau right child dari suatu node (tree tidak boleh kosong).
- Update : mengubah isi node yang ditunjuk oleh pointer current (tree tidak boleh kosong)
- Retrieve : mengetahui isi node yang di tunjuk oleh pointer current (tree tidak boleh kosong)
- DeleteSub : menghapus sebuah tree (node beserta seluruh descendantnya)
- Characteristic : mengetahui karakterisktik dari suatu tree, yakni : size, weight, serta average lengthnya. (average length = [JmlNodeLvl1*1 + JmlNodeLvl2*2 + ... + JmlNodeLvln*n]/Size)
- Traverse : mengunjungi seluruh node – node pada tree, masing – masing sekali. Ada 3 cara traverse :
    1. Pre Order
       Cetak isi node yang dikunjungi, kungjungi Left Child, kunjungi Right Child
    2. In Order
       Kunjungi Left Child, cetak isi node yang dikunjungi, kunjungi Right Child
    3. Post Order
       Kunjungi Left Child, kunjungi Right Child, cetak isi node yang dikunjungi

Soal :

1. Insert(Root,65)
2. Insert(RightChild,5)
3. Insert(LeftChild44)
4. Find(Root)
   Insert(LeftChild,31)
5. Insert(LeftChild,7)
6. Find(Root)
   Find(RightChild)
   Insert(RightChild,12)