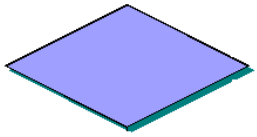
MODUL PASCAL

Mengenal Flowchart

Diagram alur *(flowchart)* merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Bagan ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu sedangkan hubungan antar proses digambarkan dengan garis penghubung. Simbol-simbol diagram alur yang digunakan penulis dalam skripsi ini diperlihatkan pada tabel berikut ini.

Simbol Diagram Alur Beserta Fungsinya

Simbol	Nama	Fungsi
	Simbol input / output	Digunakan untuk mewakili proses keluar masuknya informasi pada sistem.
	Simbol keputusan/ percabangan	Digunakan untuk mewakili proses keputusan yang dilakukan dalam sistem.
	Simbol tampilan	Digunakan untuk mewakili proses penampilan output pada suatu terminal.

	Simbol proses	Digunakan untuk menunjukkan aktifitas utama/ proses pada sistem.
	Simbol terminal	Digunakan untuk menunjukkan awal mulai dan akhir dari kegiatan.
	Simbol penghubung	Bila flowchart terpotong dan masih mempunyai sambungan.

Penggunaan Turbo Pascal

Dari modus prompt DOS, hal yang dilakukan sbb:

A:\> turbo

Menu dalam Pascal :

a. File (Alt-F)

Load / Pick : untuk mengambil program yang sudah ada di disk kerja.
New : untuk membuat program baru.
Save : untuk menyimpan program.
Write to : untuk merekam program ke suatu file.
Directory : untuk menampilkan directory.
Change dir : untuk mengganti direktory yang aktif.
OS Shell : untuk menjalankan perintah-printah DOS
Quit : mengakhiri turbo Pascal dan kembali keprompt DOS

b. Edit (Alt-E)

Digunakan untuk keperluan memperbaiki program.

c. Run (Alt-R)

Digunakan untuk menjalankan program yang ada dijendela edit.

d. Compile (Alt-C)

Digunakan untuk mengkompilasi program.
Destination Memory (disimpan di memory).
Destination Disk (disimpan di disk dengan ext .EXE).

e. Options (Alt-O)

Digunakan untuk mengatur/menentukan kembali bagaimana F1-help, F2-Save F3-new file, F4-import data, F9-expand, F10-contract dan Esc-exit integrated environment bekerja.

f. Debug dan Break/Watch (Alt-D & Alt-B)

Digunakan untuk melacak program.mengaktifkan Debug & Break/Watch.

***Cat : tekan Esc untuk meninggalkan menu.*

Struktur Program Pascal

Secara ringkas, struktur suatu program Pascal dapat terdiri dari :

1. Judul Program
2. Tubuh Program

Tubuh program dibagi menjadi dua bagian utama :

a. Bagian deklarasi
   - deklarasi label
   - deklarasi konstanta
   - deklarasi tipe
   - deklarasi variabel/perubah
   - deklarasi prosedur
   - deklarasi fungsi

b. Bagian Pernyataan/Terproses

Cat : baris-baris komentar untuk memperjelas program diletakkan diantara tanda (dan *) atau { dan } .*

1. Judul program

Judul program ini digunakan untuk memberi nama program dan sifatnya optional. Jika ditulis harus terletak pada awal dari program dan diakhiri dengan titik koma (;).

Contoh penulisan judul program :

```
PROGRAM latihan;
PROGRAM latihan(input,output);
PROGRAM lat_1;
PROGRAM lat_satu(output);
```

2. Bagian Pernyataan/Terproses

Bagian ini adalah bagian yang akan terproses dan terdapat dalam suatu blok yang diawali dengan BEGIN dan diakhiri dengan END (penulisan END diikuti dengan tanda titik).

Bagian ini berisi pernyataan / statamen yang merupakan instruksi program. Setiap statemen diakhiri dengan tanda titik koma (;).

Bentuk umumnya adalah sbb :

```
BEGIN
   ...
   statemen;
   statemen;
   ...
END.
```

3. Bagian deklarasi

Bagian ini menjelaskan / memperkenalkan secara rinci semua data yang akan digunakan pada suatu program. Dalam penulisannya tidak boleh sama dengan kata-kata cadangan (reserved words) dan selalu diakhiri dengan titik koma (;).

Deklarasi label

Deklarasi label digunakan Jika pada penulisan program akan menggunakan statemen GOTO (untuk meloncat ke suatu statement tertentu).

Contoh :

```
PROGRAM cetak;
   LABEL satu,akhir;
   BEGIN
      WRITELN('STMIK');
      GOTO SATU;
      WRITELN('BINA');
      satu:
      WRITELN('GUNADARMA');
      GOTO akhir;
      WRITELN('SARANA');
      akhir:
   END.
```

Bila program di atas dijalankan, output sbb :

```
STMIK
GUNADARMA
```

Deklarasi konstanta

Deklarasi ini digunakan untuk mengidentifikasikan data yang nilainya sudah ditentukan dan pasti, tidak dapat dirubah dalam program.

Contoh :

```
PROGRAM CETAK_2(OUTPUT);
  CONST a = 50;  (* selalu menggunakan tanda = *)
    b = 'INDONESIA Merdeka';
  BEGIN
    WRITELN(a,' TAHUN ');
    WRITELN(b);
  END.
```

Bila program dijalankan, output sbb :

```
50 tahun
INDONESIA Merdeka
```

Deklarasi tipe

Deklarasi ini digunakan untuk menyebutkan tipe setiap data yang akan digunakan pada program Pascal. Tipe data menentukan jangkauan nilai yang mungkin dari data yang digunakan

Contoh :

```
PROGRAM SATU;
TYPE bulat  = INTEGER;  { selalu  menggunakan = }
     hasil,pecahan = REAL;
     ket          = STRING[20];
BEGIN
     pecahan := 2.52;
     bulat := 2;
     hasil := pecahan + bulat;
     ket := 'hasil penjumlahan = ';
WRITE(ket,hasil:4:2);
END.
```

Output program, sbb : hasil penjumlahan = 4.52

Deklarasi variabel/perubah

Deklarasi ini berisi data-data yang bisa berubah-ubah nilainya di dalam program. Deklarasi variabel harus di letakkan setelah deklarasi tipe (jika ada).

Contoh :

```
VAR satu : INTEGER;
     dua  : INTEGER;
     a    : REAL;
     b    : REAL;        { selalu menggunakan : }
BEGIN
     satu := 5;
     dua := 4;
     a := 2.3;
     b := 5+4*2.3;       { hasil real }
WRITE('hasil = ',b:4:1);
```

```
END.
```

Output program : hasil = 14.2

Program diatas bisa ditulis sbb :

```
VAR satu,dua : INTEGER;
      a,b    : REAL;
BEGIN
   ...
   statement;
   ...
END.
```

Contoh jika terdapat deklarasi tipe :

```
TYPE
        bilangan = integer;
VAR
        satu,dua,a      : bilangan;
        b               : real;
BEGIN
   ...
   statement;
   ...
END.
```

Deklarasi prosedur dan Fungsi

Program dapat dibagi menjadi beberapa bagian/subprogram, yang terdiri dari satu program utama dan satu / lebih program bagian (bisa berupa prosedur / fungsi). Deklarasi prosedure/ fungsi terletak pada subprogram yang menggunakannya.

Tipe Data

Tipe Data dapat terletak pada deklarasi variabel maupun padadeklarasi tipe. Pascal menyediakan beberapa macam tipe data, yang terdiri dari :

1. Tipe data sederhana/skalar, terdiri dari :
   - 1.1. Tipe data standar/predefinisi
     - 1.1.1. bulat (integer)
     - 1.1.2. real
     - 1.1.3. karakter
     - 1.1.4. string
     - 1.1.5. logika (boolean)
   - 1.2. Tipe data didefinisikan pemakai :
     - 1.2.1. subjangkauan (subrange)
     - 1.2.2. terbilang (enumerated)
2. Tipe data terstruktur, terdiri dari :

2.1. larik (array)
2.2. rekaman (record)
2.3. berkas (file)
2.4. himpunan (set)
3. Tipe data penunjuk (pointer)

1.1. Tipe data standar

1.1.1. Tipe data integer

Tipe integer adalah bilangan yang tidak mempunyai titik desimal/bilangan pecahan. Integer terdiri dari beberapa tipe, yaitu :

- byte, dengan jangkauan nilai 0..255
- shortint, dengan jangkauan nilai -128..127
- integer, dengan jangkauan nilai -32768..32767
- word, dengan jangkauan nilai 0..65535
- longint, dengan jangkauan nilai -2147483648..2147483647

Operator yang dapat digunakan pada data tipe integer :

+, penjumlahan
-, pengurangan
*, perkalian
div, pembagian
mod, sisa pembagian

Contoh :

```
VAR a,b,jumlah1,jumlah2 : INTEGER;
BEGIN
   jumlah1:=10;
   jumlah2:=3;
   a:=jumlah1 DIV jumlah2;
   b:=jumlah1 MOD jumlah2;
   WRITELN('HASIL A = ',a);
   WRITELN('HASIL B =',b);
END.
```

hasil program : hasil a = 3
hasil b = 1

1.1.2. Tipe data real

Tipe real adalah bilangan yang mengandung pecahan, palingsedikit harus ada satu digit sebelum dan sesudah titik desimal.

Operator yang dapat digunakan pada data tipe real adalah :

+ penjumlahan
- pengurangan
* perkalian
/ pembagian

Contoh :

```
VAR nilai1,nilai2,hasil : REAL;
BEGIN
   nilai1 := 2.52;
   nilai2 := 3.2;
   hasil := nilai1 + nilai2;
   WRITE('HASIL PENJUMLAHAN = ',hasil:4:2);
END.
```

Output program, sbb : hasil penjumlahan = 5.72

1.1.3. Tipe data karakter

Nilai data karakter berupa sebuah karakter yang ditulis diantara tanda petik tunggal, misalnya : 'A', 'b', '@', dan sebagainya. Karakter yang dapat diterima oleh komputer :

huruf besar/kecil : A,B,C,...,Z / a,b,...,z
digit : 1,2,3,...,9
operator aritmatika : * / + -
tanda baca : , . ; : ? !
simbol khusus : $ @ { } ( ) [ ] % #
spasi

Contoh :

```
VAR nilai : CHAR;
BEGIN
   nilai :='A';
   WRITELN('NILAI TERBAIK = ',nilai);
END.
```

hasilnya : nilai terbaik = A

1.1.4. Tipe data string

Nilai data string adalah satu atau lebih karakteryang terletak diantara tanda petik tunggal, misal : 'GUNADARMA'. Bila panjang dari suatu string di dalam deklarasi variabel tidak disebutkan, maka dianggap panjangnya 255 karakter.

Contoh :

```
VAR kata1 : STRING[5];
   kata2 : STING[9];
   kata : CHAR;
BEGIN
   kata1 :='STMIK';
   kata2 :='GUNADARMA';
   kata :=' ';              { karakter berupa spasi }
   WRITELN(kata1,kata,kata2);
END.
```

hasil : STMIK GUNADARMA

1.1.5. Tipe data boolean

Data tipe boolean mempunyai dua nilai, yaitu True dan False.

Contoh :

```
VAR
  benar : BOOLEAN;
BEGIN
   benar := TRUE;
   WRITELN('benar = ',benar);
END.
hasil :   benar = TRUE
```

1.2. Tipe data Terdefinisi

1.2.1 Tipe data subjangkauan

Tipe data ini adalah tipe data yang dapat didefinisikan sendiri oleh pemakai. Nilai data pada tipe ini mempunyai jangkauan tertentu.

Misalkan nilai ujian mempunyai harga 0 sampai 100, maka nilai ujian dapat didefinisikan sbb :

```
TYPE
   nilai = 0..100;
```

Contoh :

```
VAR sks     : 1..4;
    angkatan : 89..95;
    nilai    : 'A'..'E';
```

1.2.2. Tipe data terbilang

Tipe data ini juga dapat didefinisikan sendiri oleh pemakai. Disebut tipe terbilang karena semua nilai disebut satu persatu.

Contoh :

```
TYPE hari      =  (Senin,Selasa,Rabu,Kamis,Jum'at,Sabtu,Minggu);
   hari_kerja  =  (Senin,Selasa,Rabu,Kamis,Jum'at);
   situasi     = (senang,gembira,sedih,susah);
```

2.1. Tipe data larik (array)

Larik (array) adalah kumpulan data yang mempunyai tipe data sejenis. Daftar nomor telpon, daftar kode mata kuliah, vektor, matrik merupakan contoh larik.

Contoh penulisan tipe larik berdimensi satu sbb :

```
CONST batas = 20;
VAR telpon : ARRAY[1..3] OF STRING[7]; { larik dengan nama telpon
                                         mempunyai 3 data dengan tipe
                                         string }
```

```
nilai   : ARRAY[1..5] OF INTEGER;      { larik dengan nama nilai
                                         mempunyai 5 data dengan tipe
                                         integer }
gaji  : ARRAY[1..batas] OF REAL; { larik dengan namagaji mempunyai
                                   20 data dengan tipe real }
```

Contoh larik yang mempunyai tipe data terbilang atau subjangkauan :

```
TYPE batas   = 0..100;
     keadaan = (baru,lama,bagus,jelek);
VAR nilai : ARRAY[1..30] OF 'A'..'B';  { larik dengan nama nilai mempunyai
                                         30 data, dan pengisian data yang
                                         diperbolehkan hanya A, B, C, D,
                                         E }
    angka   : ARRAY[1..50] OF batas;   { larik dengan nama angka
                                         mempunyai 50 data, dan pengisian
                                         data yang diperbolehkan hanya
                                         1,2,3,...,99,100 }
    baju  : ARRAY[1..10] OF keadaan;   { larik dengan nama angka
                                         mempunyai 10 data,dan pengisian
                                         data yang diperbolehkan baru, lama,
                                         bagus, jelek }
```

Contoh program :

```
VAR jumlah : INTEGER;
   nilai   : ARRAY[1..3] OF 'A'..'E';
  angka  : ARRAY[1..3] OF INTEGER;
BEGIN
   nilai[1] := 'C';
   nilai[2] := 'B';
   nilai[3] := 'A';
   angka[1] := 75;
   angka[2] := 60;
   angka[3] := 90;
   jumlah := angka[1]+angka[2]+angka[3];
   WRITELN('NILAI = ',angka[2],' MENDAPAT ',nilai[1]);
   WRITELN('JUMLAH = ',jumlah);
END.
```

hasil : nilai 60 mendapat C
jumlah = 225

Contoh penulisan tipe larik berdimensi dua sbb :

```
VAR tabel : ARRAY[1..3,1..2] OF BYTE;  { larik tabel mempunyai 3 baris dan 2
                                         kolom dengan tipe byte }
BEGIN tabel[1,1] := 5;   { baris 1, kolom 1 }
    tabel[1,2] := 7;
```

```
    tabel[2,1] := 21;   { baris 2, kolom 1 }
    tabel[2,2] := 18;
    tabel[3,1] := 8;
    tabel[3,2] := 7;
    WRITELN('BARIS 1 KOLOM 2 = ',tabel[1,2]);
END.
```
hasil : BARIS 1 KOLOM 2 = 7

2.2. Tipe data record dan file

(dibahas pada pembahasan record dan file)

SOAL-SOAL :

1. Apa yang kamu ketahui tentang pascal ? jelaskan !
2. Sebutkan & jelaskan secara rinci deklarasi-deklarasi dalam pascal !
3. Sebutkan langkah-langkah pembuatan program pascal !
4. Terangkan langkah-langkah penyimpanan,pemanggilan, pengeditan, dan menjalankan program pascal !
5. Bagaimana cara mengcompile program pascal menjadi program .EXE!
6. Buat program untuk menampilkan Nama, dan NPM anda !

Statemen-statemen Pada Pascal

RESERVED WORD

Reserved Word adalah kata-kata baku yang digunakan dalam program dan mempunyai bentuk serta kegunaan tertentu yang telah didefinisikan oleh Pascal.

Reserved Word tidak boleh didefinisikan kembali oleh pemakai, sehingga tidak dapat digunakan sebagai pengenal (Identifier). Dalam bahasa pemrograman Pascal, beberapa Reserved Word tersebut adalah :

AND	DOWNTO	IN	OF	STRING
ASM	ELSE	INHERITED	OR	THEN
ARRAY	END	INLINE	PACKED	TO
BEGIN	EXPORTS	INTERFACE	PROCEDURE	TYPE
CASE	FILE	LABEL	PROGRAM	UNIT
CONST	FOR	LIBRARY	RECORD	UNTIL
CONSTRUCTOR		FUNCTION	MOD	REPEAT
DESTRUCTOR		GOTO	NIL	SET
DIV	IF	NOT	SHL	WHILE
DO	IMPLEMENTATION	OBJECT	SHR	
WITH	VAR	USES		

Selain dari Reserved Word di atas, Turbo Pascal masih memiliki tambahan Reserved Word berikut :

ABSOLUTE	ASSEMBLER()	FAR	FORWARD	INDEX

BEBERAPA STATEMEN/PERINTAH PADA PASCAL

Statemen adalah perintah untuk pengerjaan program pascal. Statemen terletak di bagian deklarasi statemen dengan diawali oleh kata cadangan BEGIN dan diakhiri dengan kata cadangan END. Akhir dari setiap statemen diakhiri dengan titik koma(;). Statemen-statemen dalam bahasa Pascal terdiri dari pernyataan yang berupa fungsi dan prosedur yang telah disediakan sebagai perintah standar Turbo Pascal.

1. Statemen-statemen yang digunakan untuk input/output

1.1. READ/READLN(prosedur)

Digunakan untuk memasukkan (input) data lewat keyboard ke dalam suatu variabel.

Sintaks: READ/READLN(V);

Keterangan :

V = variabel.

READ = pada statemen ini posisi kursor tidak pindah ke baris selanjutnya.

READLN = pada statemen ini posisi kursor akan pindah ke baris selanjutnya setelah di input.

1.2. READKEY(fungsi)

Untuk pembacaan sebuah karakter dari keyboard. Tipe data yang dihasilkan adalah char.

Sintaks: READKEY;

1.3. WRITE/WRITELN(prosedur)

Digunakan untuk menampilkan isi dari suatu nilai variabel di layar.

Sintaks: WRITE/WRITELN(V);

Keterangan :

V = variabel.

WRITE/WRITELN = sama dengan READ/READLN.

Contoh :

```
PROGRAM in_out;
USES CRT;
VAR nm  : STRING;
   npm : STRING;
BEGIN
    CLRSCR;
    WRITELN('masukkan nama dan NPM ');
    WRITELN('-----------------------------');
    WRITE('nama anda : ');
    READLN(nm);
    WRITELN('NPM anda : ');
    READLN(npm);
END.
```

Bila dijalankan hasilnya adalah:

```
masukkan nama dan NPM
-----------------------------
nama anda : ( di input )
NPM anda : ( di input )
```

2. Statemen-statemen yang digunakan untuk pengaturan letak di layer

2.1. CLRSCR(prosedur)

Digunakan untuk membersihkan layar.

sintaks: CLRSCR;

2.2. GOTOXY(prosedur)

Untuk menempatkan posisi kursor pada layar.

Sintaks: GOTOXY(X, Y: Byte);

Keterangan :

X = sumbu X (posisi horisontal), Y = sumbu Y (posisi vertikal)

2.3. DELLINE(prosedur)

Untuk menghapus sebuah baris pada posisi kursor dan menaikkan baris-baris dibawahnya.

Sintaks: DELLINE;

2.4. INSLINE(prosedur)

Untuk menyisipkan sebuah baris pada posisi kursor dan menggeser kebawah tampilan-tampilan baris dibawahnya.

Sintaks: INSLINE;

2.5. DELAY(prosedur)

Untuk menghentikan sejenak proses program.

Sintaks: DELAY(MS: Word);

Keterangan : MS = ukuran waktu dalam milisecond.

Contoh :

```
PROGRAM LAYAR;
USES CRT;
VAR   x : CHAR;
BEGIN
   CLRSCR;
   GOTOXY(35,10);WRITELN('STMIK GUNADARMA');
   WRITE(tunggu sebentar...!!');
   DELAY(5000);
   INSLINE;
   GOTOXY(35,11);WRITELN('Top Banget Dech ...');
   GOTOXY(01,13);WRITELN('Tekan Enter !');
   DELAY(1000);
   GOTOXY(15,12);
   DELLINE;
   READ(x);
END.
```

Hasilnya adalah :

```
                    STMIK GUNADARMA
                    Top Banget Dech ...
   tunggu sebentar...!
   Tekan Enter !
```

3. Statemen yang digunakan untuk memanipulasi string

3.1. CONCAT(fungsi)

Untuk menggabungkan 2 atau beberapa variabel string.

Sintaks: CONCAT(s1 [,s2,...,sn]: String) : STRING;

contoh: CONCAT('ABC','DEF') { ABCDEF }

3.2. COPY(fungsi)

Mengambil satu(1) atau beberapa karakter dari sebuah string.

Sintaks: COPY(S,Index,Count) : String;

Keterangan :

S = sebuah string (string).

Index = posisi awal kita akan mengambil beberapa karakter (integer)

Count = banyaknya karakter yang akan diambil (integer).

3.3. DELETE(prosedur)

Menghapus sebagian karakter dari sebuah string.

Sintaks: DELETE(S,Index,Count);

Keterangan : sama dengan statemen COPY.

3.4. INSERT(prosedur)

Menyisipkan satu(1) atau beberapa karakter ke dalam sebuah string.

Sintaks: INSERT(Source,var S,Index);

Keterangan :

Source = sumber string untuk disisipi (string)

var S = string tujuan yang akan disisipi oleh string Source (string)

Index = posisi mulai (integer).

3.5. LENGTH(fungsi)

Memberikan nilai panjang dari suatu string (jumlah karakterdalam string).

Sintaks: LENGTH(S);

Keterangan :

S = string

LENGTH(S) menghasilkan nilai integer.

3.6. POS(fungsi)

Mencari posisi sebuah bagian string (substring) didalam sebuah string.

Sintaks: POS(Substr,S); {menghasilkan nilai Byte}

Keterangan :

Substr = substring yang akan dicari posisinya di dalam sebuah string S. Bila bernilai 0 berarti nilai string yang dicari tidak ada.

3.7. STR(prosedur)

Merubah nilai numerik ke dalam nilai string.

Sintaks: STR(N,S);

Keterangan :

N = data tipe integer,

S = data tipe string.

3.8. VAL(prosedur)

Merubah nilai string ke dalam nilai numerik.

Sintaks: VAL(S,N,P);

Keterangan :

S = nilai string,
N = nilai real,
P = posisi salah.

Nilai string harus berisi angka, plus atau minus, bila tidak berarti kesalahan dan letak kesalahannya ditunjukkan oleh variabel posisi salah. Jika benar, maka nilai variabel tsb = 0 (nol).

3.9. UPCASE(fungsi)

Memberikan huruf kapital dari argumen.

Sintaks: UPCASE(S);

Keterangan :

S = variabel bertipe karakter.

Contoh :

```
PROGRAM mani_string;
USES CRT;
VAR  s : STRING;
       l : INTEGER;
       h : STRING;
CONST  a='STMIK';
          b='STIE ';
          c='GUNADARMA';
BEGIN
   CLRSCR;
   s:=CONCAT(a,b,c);
   WRITELN(s);
   INSERT(' & ',s,6);
   WRITELN(s);
   DELETE(s,7,7);
   WRITELN(s);
   h:=COPY(s,1,5);
   WRITELN(h);
   l:=LENGTH(s);
   WRITELN('Panjangnya string S : ',l);
   WRITELN('Posisi "GUNA" pada nilai S : ',POS('GUNA',s));
END.
```

Hasilnya adalah :

```
STMIKSTIE GUNADARMA
STMIK & STIE GUNADARMA
STMIK GUNADARMA
STMIK
Panjangnya string S : 15
Posisi "GUNA" pada nilai S : 7
```

4. Statemen-statemen untuk perhitungan aritmatik

4.1. ABS(fungsi)

Memberikan nilai mutlak dari suatu argumen.

Sintaks: ABS(x);

4.2. ARCTAN(fungsi)

Memberikan nilai dari fungsi arctangent dari perhitungan goniometri.

Sintaks: ARCTAN(x);

Dimana x dapat bertipe real atau integer dan akan menghasil kan nilai bertipe real.

4.3. COS(fungsi)

Memberikan nilai dari fungsi Cosinus.

Sintaks: COS(x);

4.4. EXP(fungsi)

Menghitung nilai pangkat dari bilangan e (bilangan alam),

yaitu sebesar x.

Sintaks: EXP(x);

x dapat bertipe real atau integer dan akan menghasilkan nilai bertipe real.

4.5. FRAC(fungsi)

Untuk mendapatkan nilai pecahan dari suatu bilangan.

Sintaks: FRAC(x);

Tipe dari x sama seperti yang diatas.

4.6. INT(fungsi)

Memberikan nilai integer (bilangan bulat) dari suatu variabel dengan membuang bilangan di belakang koma.

Sintaks: INT(X);

4.7. LN(fungsi)

Digunakan untuk menghitung nilai logaritma alam (natural logarithm) dari nilai x.

Sintaks: LN(x);

4.8. SIN(fungsi)

Memberikan nilai dari fungsi Sinus.

Sintaks: SIN(x);

4.9. SQR(fungsi)

Digunakan untuk menghitung nilai pangkat kuadrat dari suatu bilangan.

Sintaks: SQR(x);

Tipe dari x bisa berupa real maupun integer. Dan hasilnya akan sama dengan tipe dari x.

4.10. SQRT(fungsi)

Digunakan untuk menghitung nilai akar dari suatu bilangan.

Sintaks: SQRT(x);
Contoh :

```
PROGRAM Aritmatik;
USES CRT;
VAR x : REAL;
BEGIN
  CLRSCR;
  WRITE('masukkan nilai dari X = ');
  READLN(x);
  IF x<0 THEN x:=ABS(x);
  WRITELN('Nilai X = ',x:5:2);
  WRITELN('Nilai eksponentialnya = ',EXP(x):9:3);
  WRITELN('Nilai logaritma alamnya = ',LN(x):9:3);
  WRITELN('Nilai integernya = ',INT(x):5:2);
  WRITELN('Nilai fraksionalnya = ',FRAC(x):5:2);
  WRITELN('Nilai X dipangkatkan = ',SQRT(x):9:3);
  WRITELN('Nilai X diakarkan = ',SQRT(x):9:3);
  WRITE('Nilai X jika dimasukkan dalam ');
  WRITELN('fungsi SIN,COS,TANGEN : ');
  WRITELN('- Sinus = ',SIN(x):9:3);
  WRITELN('- Cosinus = ',COS(x):9:3);
  WRITELN('- Tangen = ',ARCTAN(x):9:3);
END.
```

Hasilnya :

```
masukkan nilai dari X = -2.5
Nilai X = 2.50
Nilai eksponensialnya = 12,182
Nilai logarima alamnya =  0,196
Nilai integernya = 2.00
Nilai fraksionalnya = 0.50
Nilai X dipangkatkan = 6.250
Nilai X diakarkan =  1.581


Nilai X jika dimasukkan dalam fungsi SIN,COS,TANGEN :
- Sinus = 0.598
- Cosinus =  -0.801
- Tangen = 1.190
```

5. Statemen-statemen untuk transfer nilai dari suatu variable

5.1. CHR(fungsi)

Merubah nilai dari byte ke bentuk karakter yang sesuai dengan kode ASCII.

Sintaks: CHR(x);
Keterangan : x bertipe byte

contoh : WRITELN(CHR(61);
hasilnya : a

5.2. ORD(fungsi)

Merubah nilai suatu variabel dari bentuk karakter ke bentuk longint.

Sintaks: ORD(X);
Keterangan : x bertipe char
contoh : WRITELN(ORD('B');
hasilnya : 42

5.3. ROUND(fungsi)

Membulatkan data tipe real ke data tipe longint.

Sintaks: ROUND(X);
Keterangan : Jika nilai pecahan < 0,5 maka dibulatkan kebawah.
Jika nilai pecahan > 0,5 maka dibulatkan keatas.
contoh : WRITELN('10/3 dibulatkan = ',ROUND(10/3));
hasilnya : 10/3 dibulatkan = 3

5.4. TRUNC(fungsi)

Membulatkan kebawah data tipe real ke data tipe longint.

Sintaks: TRUNC(X);
contoh :
WRITELN('20/3 dibulatkan kebawah = ',TRUNC(20/3));
hasilnya : 20/3 dibulatkan kebawah = 6

6. Statemen-statemen untuk memanipulasi data

6.1. PRED(fungsi)

Memberikan nilai sebelum nilai argumen dalam urutannya dalam ASCII.

Sintaks: PRED(x);

6.2. SUCC(fungsi)

Memberikan nilai sesudah nilai argumen dalam urutannya dalam ASCII.

Sintaks: SUCC(x);

6.3. INC(fungsi)

Menambah (increments) nilai suatu variabel.

Sintaks: INC(x,i); {i >= 1}

6.4. DEC(fungsi)

Mengurangi (decrements) nilai suatu variabel.

Sintaks: DEC(x,i); {i >=1}
Contoh :

```
PROGRAM Mani_data;
USES CRT;
TYPE
   hari = (hr0,hr1,hr2,hr3,hr4,hr5,hr6,hr7)
VAR
   urutanhr : hari;
CONST
   namahr : ARRAY[hr1..hr7] OF STRING[6]=
         ('Senin','Selasa','Rabu','Kamis',
          'Jumat','Sabtu','Minggu');
BEGIN
   WRITELN('DAFTAR NAMA HARI');
   urutanhr := hr0;
   WHILE Urutanhr < hr7 DO
   BEGIN
    urutanhr := SUCC(urutanhr);
    WRITE('hari ke ',ORD(Urutanhr):2,' adalah ');
    WRITELN(namahr[urutanhr]);
   END;
END.
```

hasilnya adalah :

```
DAFTAR NAMA HARI
hari ke  1 adalah Senin
hari ke  2 adalah Selasa
hari ke  3 adalah Rabu
hari ke  4 adalah Kamis
hari ke  5 adalah Jumat
hari ke  6 adalah Sabtu
hari ke  7 adalah Minggu
```

7. Statemen-statemen tambahan (warna,suara dan window)

7.1. TEXTCOLOR(prosedur)

Untuk mengatur warna dari karakter-karakter di layar.

Sintaks: TEXTCOLOR(color : Byte);

Catatan : untuk pilihan warna lihat pada buku Turbo Pascal.

7.2. TEXTBACKGROUND(prosedur)

Untuk mengatur warna latar belakang dari karakter-karakter dilayar.

Sintaks: TEXTBACKGROUND(Color : Byte);

7.3. WINDOW(prosedur)

Untuk membuat suatu jendela (window) yang terletak pada layar.

Sintaks: WINDOW(x1,x2,y1,y2 : Byte);

x1,x2 = kordinat kiri atas dengan nilai maksimal sesuai dengan mode layar.

y1,y2 = kordinat kanan bawah dgn nilai maksimal sesuai dengan mode layar.

7.4. TEXTMODE(prosedur)

Untuk mengatur lebar layar, 80 kolom atau 40 kolom.

Sintaks: TEXTMODE(Mode: Byte);

Default = C80

7.5. SOUND(prosedur)

Untuk mengaktifkan suara(beep) pada internal speaker.

Sintaks: SOUND(Hz : word);

Untuk mengnonaktifkannya, gunakan statemen NOSOUND.

Contoh :

```
PROGRAM Layar2;
USES CRT;
BEGIN
   CLRSCR;
   WINDOW(5,5,20,75);
   TEXTBACKGROUND(RED);
   TEXTCOLOR(YELLOW);
   SOUND(220);
   GOTOXY(10,7);
   WRITELN('Laboratorium Komputer');
   GOTOXY(11,7);
   WRITELN('Manejemen Informatika');
   NOSOUND;
END.
```

SOAL-SOAL :

Soal I :

Buatlah program dibawah ini dengan tampilan menggunakan perintah Window, Textcolor, Textbackground, Gotoxy, dan Sound untuk memperindah tampilan.

1. Mengubah derajat temperatur, dari derajat Celcius ke derajat Fahreinheit dan Reamur (derajat Celcius diinput
2. Menghitung Luas dan Keliling lingkaran, dengan jari-jari diketahui (diinput).
3. Menghitung Luas dan Keliling segitiga sembarang yang diketahui ke tiga sisinya.
4. Mencari nilai Sinus, Cosinus, dan Tangen dengan sudut diinput.
5. Mencari akar dan kuadrat dari suatu nilai (nilai diinput).
6. Mencari nilai bulat dan pecahan dari suatu nilai yang dimasuk kan melalui keyboard (diinput). Nilai pecahan tersebut dibulatkan sampai 3 angka dibelakang koma (,).
7. Tampilkan nama dan NPM anda di dalam window, dan terletak pada tengah-tengah layar.
8. Tampilkan tulisan 'STMIK GUNADARMA' di dalam window pada pojok kanan atas dengan ukuran window sama dengan tulisan tersebut.

Soal II :

Buatlah program pada soal jenis I (no. 1-6) dengan tampilan menggunakan 2 window. Window yang pertama digunakan untuk nilai yang diinput. Window yang kedua untuk hasil dari program (output).

Soal III :

1. Buatlah program untuk menggabungkan 2 buah kata yang diinput. Setiap kata yang diinput harus berada didalam window yang dan hasilnya berada pada window yang berbeda pula.
2. Buatlah program untuk menampilkan window secara acak dengan warna yang berbeda.

Bentuk - Bentuk Perulangan dan Penyeleksian Kondisi

BENTUK - BENTUK PERULANGAN

Dalam hampir setiap program yang kompleks mutlak memerlukan suatu perulangan dan percabangan. Tujuan perulangan disini adalah untuk mengulang statement atau blok statement berulang kali sesuai sejumlah yang ditentukan pemakai. Dalam materi ini akan memberikan gambaran konsep dasar dari pengertian diatas.

1. Perulangan FOR

Perulangan dengan statemen FOR digunakan untuk mengulang statemen atau suatu blok statemen berulang kali. Perulangan dengan statemen FOR dapat berupa perunlangan positif dan perulangan negatif.

Perulangan FOR positif

Contoh :

Perulangan positif untuk satu statement :

```
USES CRT;
VAR
   i : INTEGER;
BEGIN
   FOR i := 1 TO 5 DO WRITELN('STMIK GUNADARMA');
END.
```

Maka bila program diatas dicompile

hasilnya :

```
STMIK GUNADARMA
STMIK GUNADARMA
STMIK GUNADARMA
STMIK GUNADARMA
STMIK GUNADARMA
```

Penjelasan : Berati statemen STMIK GUNADARMA akan diulang sebanyak 5 kali yaitu dengan menghitung nilai i dari i ke 1 sampai nilai i terakhir yaitu i ke 5.

Contoh dengan menggunakan blok statement:

Cara penulisannya dengan pada awal blok diawali dengan BEGIN dan pada akhir blok diakhiri dengan END;

```
USES CRT;
VAR
   i : INTEGER;
BEGIN
   FOR i:= 1 TO 10 DO
   BEGIN
```

```
        WRITELN('STMIK GUNADARMA');    { blok statement }
      END;
END.
```

Hasil yang akan didapat akan sama dengan contoh yang pertama, tapi yang harus diingat disini untuk penggunaan blok pada perulangan FOR biasanya mempunyai banyak statement (lebih dari 1 statement)

Contoh 3 :

Peggunaan perulangan FOR dalam blok statement untuk membuat tabel

```
USES CRT;
VAR
  a,b,c : INTEGER;
  bagi  : REAL;
BEGIN
   WRITELN('-------------------------------------------');
   WRITELN(' a      a*a      a*a*a        1/a      ');
   WRITELN('-------------------------------------------');
   FOR a:= 1 TO 10 DO
   BEGIN
    b:= a*a;
    c:=a*a*a;
    bagi := 1/a;
    WRITELN(a:4,c:10,d:10,bagi:12:3);
   END;
   WRITELN('-------------------------------------------');
END.
```

maka hasilnya :

```
-------------------------------------------
 a      a*a      a*a*a          1/a
-------------------------------------------
 1       1         1         1.000
 2       4         8         0.500
 3       9        27          0.333
 4      16         64         0.250
 5      25        125         0.200
 6      36        216         0.167
 7      49        343         0.143
 8      64        512         0.125
 9      81        729         0.111
10     100       1000          0.100
-------------------------------------------
```

Perulangan FOR negatif

Perulangan negatif adalah perulangan dengan menghitung (counter) dari besar ke kecil. Statement yang digunakan adalah FOR-DOWNTO-DO

Contoh :

```
USES CRT;
VAR
     i  : INTEGER ;
BEGIN
   FOR i := 10 DOWNTO 1 DO WRITE(i:3);
END.
```

Hasil :

```
10  9  8  7  6  5  4  3  2  1
```

Perulangan FOR tersarang

Perulangan FOR tersarang adalah perulangan FOR yang berada pada perulangan yang lainnya. Perulangan yang lebih dalam akan diproses terlebih dahulu sampai habis, kemudian perulangan yang lebih luar baru akan bertambah, mengerjakan perulangan yang lebih dalam lagi mulai dari nilai awalnya dan seterusnya.

Contoh :

```
VAR
 a,b : INTEGER;
BEGIN
   FOR  a := 1 TO 3 DO
   BEGIN
    FOR b := 1 TO 2 DO  WRITE(a :4,b:2);
    WRITELN;
   END;
END.
```

Hasil :

```
1 1    1 2
2 1    2 2
3 1    3 2
```

2. Perulangan WHILE-DO

Penyeleksian kondisi digunakan untuk agar program dapat menyeleksi kondisi, sehingga program dapat menentukan tindakan apa yang harus dikerjakan, tergantung dari kondisi yang diseleksi tersebut. Perulangan WHILE-DO tidak dilakukan jika kondisi tidak terpenuhi.

Contoh :

```
USES CRT;
VAR  i : INTEGER;
BEGIN
   i := 0;
   WHILE i < 5 do
```

```
        BEGIN
         WRITE(i:3);
         INC(i);   { sama dengan i:=i+1 }
        END;
    END.
```

Hasilnya :

```
0 1 2 3 4
```

Perulangan WHILE-DO tersarang

Perulangan WHILE-DO tersarang (nested WHILE-DO) merupakan perulangan WHILE-DO yang satu di dalam perulangan WHILE-DO yang lainnya.

Contoh :

```
USES CRT;
VAR
  a, b : INTEGER;
BEGIN
   CLRSCR;
   a:=1;
   b:=1;
   WHILE a < 4 DO  { loop selama a masih lebih kecil dari 4 }
   BEGIN
      a := a+1;
      WHILE b < 3 DO { loop selama b masih lebih kecil dari 3 }
      BEGIN
        WRITE(a:3,b:2);
         b:=b+1;
      END;
   END;
   READLN;
END.
```

3. Perulangan REPEAT-UNTIL.

REPEAT-UNTIL digunakan untuk mengulang statement-statemen atau blok statement sampai (UNTIL) kondisi yang diseleksi di UNTIL tidak terpenuhi. Sintak dari statement ini adalah :

Contoh

```
VAR
   i : INTEGER;
BEGIN
   i:=0;
   REPEAT
     i:= i+1;
     WRITELN(i);
   UNTIL i=5;
```

```
END.
```

hasil :

```
1
2
3
4
5
```

REPEAT-UNTIL tersarang

REPEAT-UNTIL tersarang adalah suatu perulangan REPEAT-UNTIL yang satu berada didalam perulangan REPEAT-UNTIL yang lainnya.

Contoh :

```
VAR
   a,b,c   : REAL;
BEGIN
   WRITELN('=======================================');
   WRITELN(' sisi A     sisi B          Sisi C   ');
   WRITELN(' =======================================');
   a:= 1;
   REPEAT          { perulangan luar }
    b := 0;
    REPEAT     { perulangan dalam }
     c:=SQRT(a*a+b*b);
     WRITELN(a:6:2, b:9:2, c:9:2);
     b:=b+5;
    UNTIL  b>25;  { berhenti jika b lebih besar  dari  5  untuk
                    perulangan dalam }
      a:=a+1;
   UNTIL  a>3;    { berhenti jika a lebih besar  dari  3  untuk
                    perulangan luar }
   WRITELN(' =======================================');
END.
```

BENTUK-BENTUK PERCABANGAN / PENYELEKSIAN KONDISI

1. IF-THEN

Bentuk struktur IF-THEN adalah sebagai berikut :

```
IF Kondisi THEN Statement
```

Ungkapan adalah kondisi yang diseleksi oleh statement IF. Bila kondisi yang diseleksi terpenuhi, maka statement yang mengikuti THEN akan diproses, sebaliknya bila kondisi tidak terpenuhi, maka yang akan diproses statement berikutnya.

Misalnya :

```
IF Pilihan = 2 THEN
```

```
BEGIN    { jika kondisi terpenuhi, Yaitu jika pilihan = 2 }
 ......
 ......
END
ELSE     { jika kondisi tidak terpenuhi, yaitu jika pilhan
         tidak sama dengan 2}
BEGIN
 .......
 .......
END;
```

Contoh Program :

```
USES CRT;
VAR
   Nilai : REAL;
BEGIN
   WRITE('Jumlah Nilai :');
   READLN(nilai);          {  Pemasukan data  }
   IF nilai >60 THEN       {  seleksi kondisi variabel nilai  }
   WRITELN('Lulus')        {  Dilaksanakan  jika  nilai  lebih besar dari 60  }
   ELSE
   WRITELN('Tidak lulus'); {  Dilaksanakan jika variabel  nilai lebih kecil dari 60  }
END.
```

Hasil :

Jika kita Memasukan 40 pada varibel nilai, Maka program diatas akan mencetak Tidak lulus.

IF tersarang (nested IF)

Struktur IF tersarang merupakan bentuk dari suatu statement IF berada di dalam lingkungan statemen IF yang lainya. Bentuk statement IF tersarang sebagai berikut :

```
IF kondisi1 THEN          atau        IF Kondisi1 THEN
IF kondisi2 THEN                      BEGIN
   statemen1                          IF kondisi2 THEN
ELSE                                    statemen1
  statemen2;                         ELSE
                                       statemen2
                                     END;
```

2. CASE-OF

Struktur CASE-OF mempunyai suatu ungkapan logika yang disebut dengan selector dan sejumlah statemen yang diawali dengan suatu label permasalahan (case label) yang mempunyai tipe sama dengan selector.

Statement yang mempunyai case label yang bernilai sama dengan case label yang bernilai sama dengan nilai selector akan diproses sedang statemen yang lainya tidak.
Bentuk struktur dari CASE-OF :

```
CASE Variabel Kondisi OF
   CASE- LABEL 1; STATEMENT 1;
   CASE- LABEL 2; STATEMENT 2;
   .....
   CASE- LABEL N; STATEMENT N;
END;                    { end dari case }
```

Daftar case label dapat berupa konstanta, range dari konstanta yang bukan bertipe real.

Contoh program ;

```
PROGRAM nilai;
VAR
    nil : CHAR;
BEGIN
   WRITE('Nilai Numerik yang didapat :');
   READLN(nil);
   CASE nil OF
     'A': WRITELN('SANGAT BAIK');
     'B': WRITELN('BAIK');
     'C': WRITELN('CUKUP');
     'D': WRITELN('KURANG');
     'E': WRITELN('SANGAT KURANG ');
   END;
END.
```

hasil :

```
Nilai Numerik yang didapat : B
BAIK
```

Contoh listing program untuk dicoba :

1. Program input data dengan array

```
PROGRAM pemakaian_Array_Untuk_10_data_dengan_menggunakan_For;
USES CRT;
CONST
  garis='-----------------------------------------------------';
VAR
   nil1,nil2 : ARRAY [1..10] OF 0..100; {Array dgn Type subjangkauan}
   npm       : ARRAY [1..10] OF STRING[8];
   nama      : ARRAY [1..10] OF STRING[15];
   n,i,bar   : INTEGER;
   jum       : REAL;
```

```
    tl       : CHAR;
BEGIN
    CLRSCR;
    { pemasukan data dalam array }
    WRITE('MAU ISI BERAPA DATA :');
    READLN(N);
    FOR i:= 1 TO n DO
    BEGIN
       CLRSCR;
       GOTOXY(30,4+1); WRITE('DATA KE-:',i:2);
       GOTOXY(10,5+i); WRITE('NPM     :'); READLN(NPM[i]);
       GOTOXY(10,6+i); WRITE('NAMA    :'); READLN(NAMA[i]);
       GOTOXY(10,7+i); WRITE('NILAI 1 :'); READLN(NIL1[i]);
       GOTOXY(10,8+i); WRITE('NILAI 2 :'); READLN(NIL2[i]);
     END;
    { proses data dalam array }
    CLRSCR;
    GOTOXY(5,4); WRITE(GARIS);
    GOTOXY(5,5); WRITE('NO');
    GOTOXY(9,5); WRITE('NPM');
    GOTOXY(18,5); WRITE('NAMA');
    GOTOXY(34,5); WRITE('NIL.1');
    GOTOXY(41,5); WRITE('NIL.2');
    GOTOXY(47,5); WRITE('RATA');
    GOTOXY(54,5); WRITE('ABJAD');
    GOTOXY(5,6); WRITE(GARIS);
    {  proses Cetak isi array dan seleksi kondisi  }
    bar:=7;
    FOR i:= 1 TO n DO
    BEGIN
       jum:=(nil1[i]+nil2[i])/2;
       IF jum>=90 THEN tl:='A'
       ELSE
          IF jum>80 THEN tl:='B'
          ELSE
             IF jum>60 then tl:='C'
             ELSE
                IF jum 50 THEN tl:='D'
                ELSE
                   tl:='E';
       {  cetak hasil yang disimpan di array dan hasil  }
       {  penyeleksian kondisi  }
       GOTOXY(5,bar); WRITELN(i:2);
       GOTOXY(9,bar); WRITELN(NPM[i]);
       GOTOXY(18,bar); WRITELN(NAMA[i]);
```

```
        GOTOXY(34,bar); WRITELN(NIL1[i]:4);
        GOTOXY(41,bar); WRITELN(NIL2[i]:4);
        GOTOXY(47,bar); WRITELN(jum:5:1);
        GOTOXY(54,bar); WRITELN(tl);
        bar:=bar+1;
    END;
    GOTOXY(5,bar+1);WRITELN(garis);
    READLN;
END.
```

2. Program jendela bergerak

```
PROGRAM Window_Bergerak_dgn_delay;
USES CRT;
VAR i : INTEGER;
BEGIN
    FOR i:=1 TO 15 DO
    BEGIN
        SOUND(i*100);
        DELAY(100);
        NOSOUND;
    END;
    TEXTBACKGROUND(black);
    CLRSCR;
    FOR i := 1 TO 9 DO
    BEGIN
        TEXTBACKGROUND(white);
        WINDOW(42-i*4,10-i,38+i*4,15+i);
        CLRSCR;
        DELAY(100);
    END;
    TEXTCOLOR(15);
    GOTOXY(28,2);WRITELN('c');
    GOTOXY(8,3); WRITELN('3');
    GOTOXY(28,4); WRITELN('A');
    TEXTCOLOR(black);
    GOTOXY(44,3); WRITELN('3');
    GOTOXY(44,2); WRITELN('&');
    GOTOXY(29,4); WRITELN('U');
    TEXTCOLOR(red*25);
    GOTOXY(30,3); WRITELN('B E L A J A R');
    TEXTCOLOR(black);
    GOTOXY(5,5); WRITE('c');
    FOR i := 6 TO 64 DO
    BEGIN
```

```
        GOTOXY(i,5);WRITELN('');
    END;
    FOR i := 6 TO 20 DO
    BEGIN
        GOTOXY(5,i); WRITELN('3 ');
    END;
    GOTOXY(5,21); WRITELN(' ');
    TEXTCOLOR(white);
    GOTOXY(65,5); WRITE('U');
    FOR i := 6 TO 65 DO
    BEGIN
        GOTOXY(i,21); WRITELN('  `);
    END;
    FOR i := 6 TO 20 DO
    BEGIN
        GOTOXY(65,i); WRITELN('3');
    END;
    GOTOXY(65,21); WRITELN('c');
    TEXTCOLOR(yellow);
    READLN;
END.
```

SOAL - SOAL :

Buatlah program untuk soal dibawah ini dengan tampilan sebagus mungkin (gunakan perintah Window, Textcolor dll). Jumlah suku sesuai dengan input dari keyboard.

1. Buat deret hitung 3,7,11,15,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
2. Buat deret ukur 3,9,27,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
3. Buat tabel deret bergoyang 1,-2,4,-8,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
4. Buat deret suku harmonis 1,1/2,1/3,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
5. Buat deret fibbonaci 1,1,2,3,5,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
6. Buat deret seperti berikut 1,-2,3,-4,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
7. Buat deret kuadrat 1, 4, 9,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
8. Buat deret seperti berikut 100, 90, 70 ,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
9. Buat deret seperti berikut 256, 196, 144,.....=?
   Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
10. Buat deret seperti berikut 1, 1, 1, 2, 2, 3, 6, 4, 24, 5.....=?
    Program akan berhenti jika pada pertanyaan "Hitung Lagi [Y/T] ?" diisi huruf T.
11. Buatlah program untuk mencari faktorial, sesuai dengan input yang diminta.
12. Buatlah program huruf yang berjatuhan sehingga membentuk suatu kalimat yang telah diinput dari keyboard .

Array

Array adalah tipe data terstruktur yang terdiri dari sejumlah komponen komponen yang mempunyai tipe sama. Komponen tersebut disebut sebagai komponen type, larik mempunyai jumlah komponen yang jumlahnya tetap. Banyaknya komponen dalam larik ditunjukkan oleh suatu index, dimana tiap komponen di array dapat diakses dengan menunjukkan nilai indexnya atau subskript.

Array dapat bertipe data sederhana seperti byte, word, integer, real, bolean, char, string dan tipe data scalar atau subrange. Tipe larik mengartikan isi dari larik atau komponen-komponenya mempunyai nilai dengan tipe data tersebut.

Contoh :

```
var
untai : array[1..50] of integer;
```

Pada contoh Array dengan nama untai telah dideklarasikan dengan tipe integer, dengan jumlah elemen maksimum 50 elemen, nilai dari elemen array tersebut diatas harus bertipe integer.

Contoh :

```
Program contoh_array_input;
uses crt;
var
bilangan : array[1..50] of integer;
   begin
   clrscr;
   bilangan[1]:=3;
   bilangan[2]:=29;
   bilangan[3]:=30;
   bilangan[4]:=31;
   bilangan[5]:=23;
   writeln('nilai varibel bilangan ke 3 =',bilangan[3]);
   readln;
  end.
```

Array juga dapat dideklarasikan bersama dengan tipe yang beragam seperti contoh dibawah ini :

```
Program contoh_deklarasi_array_beragam;
uses crt;
var
  NPM     : array[1..20] of string[10];
  nama    : array[1..20] of string[25];
  nilai   : array[1..20] of real;
  umur    : array[1..20] of byte;
  banyak,i : integer;
begin
  clrscr;
  write('Isi berapa data array');readln(banyak);
```

```
 for i := 1 to banyak do
  begin
   write('NPM  =');readln(npm[i]);
   write('Nama =');readln(nama[i]);
   write('Nilai=');readln(nilai[i]);
   write('umur =');readln(umur[i]);
  end;
 {cetak varibel array}
  writeln('NPM       NAMA              NILAI UMUR ');
 for i:= 1 to banyak do
  begin
   writeln(npm[i]:10,nama[i]:25,nilai[i]:3:2,' ',umur[i]:3);
  end;
  READLN;  end.
```

Untuk deklarasi array dapat digunakan beberapa cara seperti beri kut ini :

```
Type
  Angka =string[20];
Var
  nama : array [1..50] of angka;
begin
  .
end.
```

Deklarasi tipe indeks subrange integer

Indeks pada array dapat tipe skalar atau subrange, tetapi tidak bisa real.

Contoh:

```
 var
   nilai : array[1..10] of integer;
```

Pada contoh ini array nilai mempunyai 10 buah elemen yaitu dari 1 sampai 10. Array tersebut dapat dideklarasikan dengan type seperti berikut ini :

```
Type
  skala = 1..10;
var
  nilai   : array [skala] of integer;
atau :
Type
  skala = 1..10;
Y      = array[skala] of integer;
var
   nilai :  Y;
atau :
 Type
    Y     = array[1..10] of integer;
```

```
  var
    nilai : Y;
Atau :
const
   atas     =1;
   bawah = 5;
type
  y = array[atas..bawah] of integer;
var
   nilai : y;
```

I. Deklarasi type indeks skalar

Indeks dari larik dapat berupa tipe skalar.

Contoh :

```
program deklarasi_indeks_array_skalar;
uses crt;
var
jum : array[(jan,feb,mar,apr,mei)] of integer;
begin
jum[jan]:=25;
jum[feb]:=45;
jum[mar]:=21;
jum[apr]:=23;
jum[mei]:=50;
writeln('Jumlah nilai bulan maret =',jum[mar]);
readln;
end.
dapat juga ditulis :
type
  bln = (jan,feb,mar,apr,mei);
Var
  jum : array[bln] of integer;
atau :
type
  bln =(jan,feb,mar,apr,mei);
var
 jum : array[jan..mei] of integer;
```

II. Deklarasi konstanta array

Array tidak hanya dapat berupa suatu varibel yang dideklarasikan di bagian deklarasi variabel, tetapi dapat juga berupa konstanta (const).

Contoh :

```
program contoh_deklarasi_array_konstan;
 uses crt;
 const
```

```
  tetap : array[1..4] of integer=(7,10,21,20);
 var
  i     : integer;
 begin
  for i:= 1 to 4 do
    writeln('Nilai Konstan array ke ',i:2,' =',tetap[i]);
    readln;
 end.
```

Konstanta array dapat juga berupa ketetapan dalam bentuk karakter seperti berikut.
Contoh :

```
program contoh_konstan_array_char_;
uses crt;
 const
  huruf : array[0..5] of char=('A','B','C','D','E','F');
 VAR
  i     : integer;
 begin
  for i:= 0 to 5 do
     writeln('Nilai konstan array ke',i:2,' = ',huruf[i]);
  readln;
 end.
```

Konstanta array dapat juga berupa string seperti berikut ini.
Contoh :

```
program constanta_array_string;
uses crt;
 type
    A     = array [1..5] of string;
 const
    Nama : A = ('basic','pascal','cobol','paradox','dbase');
 var
    I : integer;
 begin
   for i:= 1 to 5 do
     writeln('Nilai array ke-',i:2,'= ',nama[i]);
     readln;
 end.
```

Dalam pascal string merupakan array dari elemen- elemen karakter seperti berikut :
Contoh :

```
program string_adalah_array_tipe_char;
uses crt;
 var
```

```
 nama : string;
 i : integer;
begin
 nama:='Turbo Pascal';
 for i:= 1 to length(nama) do
   writeln('Elemen ',i,' dari ',Nama,'= ',nama[i]);
 readln;
end.
```

Contoh program bilangan prima dengan menggunakan bantuan array.

```
program mencari_bilangan_prima_dengan_array;
uses crt;
 var
  prima : array[1..100] of integer;
  i,j      : integer;
  bil      : integer;
 begin
 clrscr;
 for i := 2 to 100 do
  begin
   prima[i]:=i;
    for j:= 2 to i-1 do
      begin
   bil := (i mod j);  { i dibagi j dicek apakah 0}
   if bil = 0 then prima[i]:=0;
        {jika habis dibagi,berarti bkn prima}
     end;
     if prima[i]<> 0 then write(prima[i],' ');
        {cetak array yg prima}
  end; readln; end.
```

Contoh pengurutan data dengan metode buble sort, yaitu dengan cara penukaran, dapat dilihat pada contoh dibawah ini :

Contoh program :

```
program penggunaan_array_untuk_sortir_buble_sort;
uses crt;
 var
  nil1 : array[1..100] of integer;
  n,i,j,dum : integer;
 begin
   clrscr;
    write('mau isi berapa data acak (integer) ='); readln(n);
     for i := 1 to n do
       begin
```

```
      Write('Data Ke ',i,':');readln(nil1[i]);
    end;

  {* penyapuan proses}
  for i:= 1 to n-1 do
    begin
    for j:= i to n do
     begin
      if nil1[j]<nil1[i] then
       begin
         dum:=nil1[j];
         nil1[j]:=nil1[i];
         nil1[i]:=dum;
       end;
    end;
end;
 writeln;
 writeln('Hasil Sortir');
 for i := 1 to n do
  write(nil1[i]:3);
readln;
end.
```

III. Array dua dimensi

Di dalam pascal Array dapat berdimensi lebih dari satu yang disebut dengan array dimensi banyak (Multidimensional array), disini akan dibahas array 2 dimensi saja. Array 2 dimensi dapat mewakili suatu bentuk tabel atau matrik, yaitu indeks pertama menunjukkan baris dan indeks ke dua menunjuk kan kolom dari tabel atau matrik.

Untuk mengetahui cara mendeklarasikan dari penggunaan aray dua dimensi dapat dilihat pada listing program dibawah ini .

Contoh :

```
Program deklarasi_array_dua_dimensi;
uses crt;
 var
   tabel : array[1..3,1..2] of integer;
   i,j   : integer;
 begin
  clrscr;
  tabel[1,1]:=1;
  tabel[1,2]:=2;
  tabel[2,1]:=3;
  tabel[2,2]:=4;
  tabel[3,1]:=5;
  tabel[3,2]:=6;
  for I := 1 to 3 do
```

```
  begin
   for J:= 1 to 2 do
    begin
     writeln('Elemen ',i,',',j,'= ',tabel[i,j]);
    end;
   end;
  readln;
 end.
```

IV. Alternatif deklarasi array dua dimensi

Ada beberapa cara dalam mendeklarasikan array dua dimensi, beberapa cara tersebut dapat dilihat dibawah ini :

Contoh :

```
Var
  tabel : array[1..3] of array[1..2] of byte;
```

atau :

```
type
  matrik = array[1..3,1..2] of byte;
var
  tabel    : matrik;
```

atau :

```
Type
   baris    = 1..3;
   kolom  = 1..2;
   matrik  = array[baris,kolom] of byte;
var
   tabel     : matrik;
```

atau :

```
type
   baris = 1..3;
   kolom=1..2;
   matrik=array[baris] of array[kolom] of byte;
var
   tabel   : matrik;
```

Dibawah ini akan diberikan listing program penggunaan array dua dimensi dalam aplikasi penjumlahan matrik :

Contoh:

```
Program Penjumlahan_matrik;
uses crt;
 var
  matrik1,matrik2
   , hasil          : array[1..3,1..2] of integer;
   i,j              : integer;
 begin
 clrscr;
```

```
{ input matrik ke satu }
writeln(' Elemen matrik satu');
 for i := 1 to 3 do
  begin
   for j := 1 to 2 do
    begin
     write('Elemen baris -',i,' kolom -',j,'= ');
     readln(matrik1[i,j]);
    end;
  end;
 {input matrik ke dua}
 writeln('input elemen matrik dua');
  for i:= 1 to 3 do
    begin
     for j:= 1 to 2 do
      begin
   write('Elemen baris -',i,' kolom -',j,'= ');
       readln(matrik2[i,j]);
      end;
    end;
  {proses penjumlahan tiap elemen}
   for i := 1 to 3 do
     begin
      for j:= 1 to 2 do
   begin
     hasil[i,j]:=matrik1[i,j]+matrik2[i,j];
   end;
      end;
  {proses cetak hasil}
  for i:= 1 to 3 do
   begin
    for j:= 1 to 2 do
     begin
   write(hasil[i,j]:6);
   end;
   writeln;
    end;
   readln;
  end.
```

V. Array sebagai parameter

Array dapat digunakan sebagai parameter yang dikirimkan baik secara nilai (by value) atau secara acuan (by reference) ke procedure atau ke function. Procedure yang menggunakan parameter berupa array harus dideklara sikan di dalam judul procedure yang menyebutkan parameternya bertipe array.

Contoh :

```
program contoh_pengiriman_parameter_array_di_procedure;
uses crt;
 const
  garis ='-------------------------------------------------';
 type
  untai   = array[1..10] of string[15];
  bulat   = array[1..10] of integer;
  huruf   = array[1..10] of char;
 var
  i,banyak   : integer;
 procedure  proses(nama:untai;nilai:bulat);
  var
   ket   : string;
   abjad : char;
  begin
  writeln(garis);
  writeln('Nama            Nilai  Abjad   Keterangan');
  writeln(garis);
   for i := 1 to banyak do
  begin
   if nilai[i] > 90 then
     begin
       abjad:='A';
       ket  :='Istimewa';
     end;
   if (nilai[i]<90) and (nilai[i]>70) then
      begin
       abjad:='B';
       ket  :='Memuaskan';
      end;
   if (nilai[i]<70) and (nilai[i]>60) then
      begin
       abjad:='C';
       ket :='Cukup';
      end;
   if (nilai[i]<60) and (nilai[i]>45) then
      begin
        abjad:='D';
        ket :='Kurang';
      end;
   if nilai[i]< 45 then
     begin
      abjad:='E';
            ket  :='Sangat kurang';
```

```
      end;
writeln(nama[i]:15,' ',nilai[i]:4,'    ',abjad,'   ',ket:15);
 end;
     writeln(garis);
end;
procedure masuk_data;
   var
    nama  : untai;
    nilai : bulat;
   begin
    write('banyak data =');readln(banyak);
     for i:= 1 to banyak do
      begin
   clrscr;
   writeln('Data ke - ',i);
   write('Nama =');readln(nama[i]);
   write('Nilai =');readln(nilai[i]);
     end;
    proses(nama,nilai);
   end;
   {modul Utama}
   begin
     masuk_data;
     readln;
   end.
```

Record

Tipe data record merupakan tipe data terstruktur. Dalam penggunakan tipe data record dapat dikumpulkan beberapa item data yang masing- masing mempunyai tipe data berbeda- beda. Record dapat berisi beberapa field untuk sebuah subyek tertentu.

I. Deklarasi record

Diawali kata cadangan Record , lalu diikuti daftar field dan diakhiri kata cadangan end;

Contoh :

```
type
  data_pegawai = record
                   kd_peg : string[5];
                   nama   : string[15];
                   alamat : string[20];
                   gaji   : longint;
                 end;
  var
   pegawai : data_pegawai;
```

atau langsung di deklarasikan di varibel :

```
var
  pegawai : record
              kd_peg : string[5];
              nama   : string[15];
              alamat : string[20];
              gaji   : longint;
            end;
```

Cara menggunakan tiap field dari record untuk input, cetak dan proses adalah sebagai berikut :

```
Nama_record.Nama_field
```

Contoh :

```
program contoh_record_sederhana;
uses crt;
 type
  data_pegawai = record
              kd_peg : string[5];
              nama   : string[15];

              alamat : string[20];
              gaji   : longint;
            end;
  var
   pegawai : data_pegawai;
  begin
   clrscr;
   write('Kode pegawai   =');readln(pegawai.kd_peg);
   write('Nama pegawai   =');readln(pegawai.nama);
   write('Alamat pegawai =');readln(pegawai.alamat);
   write('Gaji pegawai   =');readln(pegawai.gaji);
   {cetak}
   writeln('Kode pegawai   :',pegawai.kd_peg);
   writeln('Nama pegawai   :',pegawai.nama);
   writeln('Alamat pegawai :',pegawai.alamat);
   writeln('Gaji pegawai   :',pegawai.gaji);
   readln;
  end.
```

II. Statemen with

Penggunaan statemen nama_record.nama_field seperti contoh sebelumnya dapat diringkas menjadi :

Contoh :

```
program contoh_record_menggunakan_statmen_with;
```

```
uses crt;
 type
 data_pegawai = record
                  kd_peg : string[5];
                  nama   : string[15];
                  alamat : string[20];
                  gaji   : longint;
                end;
  var
    pegawai : data_pegawai;
  begin
   clrscr;
    with pegawai do
     begin
      write('Kode pegawai   =');readln(kd_peg);
      write('Nama pegawai   =');readln(nama);
      write('Alamat pegawai =');readln(alamat);
      write('Gaji pegawai   =');readln(gaji);
      {cetak}
      writeln('Kode pegawai   :',kd_peg);
      writeln('Nama pegawai   :',nama);
      writeln('Alamat pegawai :',alamat);
      writeln('Gaji pegawai   :',gaji);
     end;
    readln;
 end.
```

Penjelasan :
Dengan menggunakan staremen with maka blok statemen berikutnya setelah statemen With dapat menggunakan nama field tanpa menyebut kan nama recordnya lagi.

III. Record dalam array

Dalam contoh sebelunya penggunan tipe data record hanya dapat menyimpan satu record. Untuk dapat menyimpan sejumlah record maka dapat digunakan array yang bertipe record yang sudah didifinisikan. Untuk itu dapat dilihat listing program berikut.
Contoh :

```
program contoh_record_dalam_array;
uses crt;
 type
  data_pegawai = record
                  kd_peg : string[5];
                  nama   : string[15];
                  alamat : string[20];
                  gaji   : longint;
                 end;
```

```
var
  pegawai     : array[1..10] of data_pegawai;
  i               : integer;
begin
 clrscr;
 for I:= 1 to 10 do
  begin
   with pegawai[i] do
    begin
 writeln('Record ke- ',i);
 write('Kode pegawai   =');readln(kd_peg);
 write('Nama pegawai   =');readln(nama);
 write('Alamat pegawai =');readln(alamat);
 write('Gaji pegawai   =');readln(gaji);
 writeln;
 end;
    end;
    {cetak}
   writeln('Kode pegawai    Nama     Alamat      gaji');
   for i:= 1 to 10 do
    begin
 with pegawai[i] do
 begin
  write(kd_peg:5);
  write(nama:15);
  write(alamat:20);
  writeln(gaji:10);
 end;
end;
    writeln('----------------------------------------');
 readln;
end.
```

IV. Field record bertipe array

Jika dalam suatu record terdapat beberapa field yang sama tipenya dapat digunakan array. Contoh ada data barang yang mempunyai struktur.

- Nama barang -> bertipe String
- Jumlah unit barang ke 1 -> bertipe Byte
- Jumlah unit barang ke 2-> bertipe Byte
- Jumlah unit barang ke 3-> bertipe Byte

Terlihat bahwa jumlah unit barang 1,2,3 bertipe sama. Dalam hal ini dapat digunakan array ber index 1.. 3 untuk mempersingkat filed jumlah unit barang.

Contoh :

```
program penggunaan_field_record_tipe_array;
```

```
uses crt;
type
 data_brg = record
        namaBrg : string[15];
        unitBrg : array[1..3] of byte;
       end;
var
  Barang        : array[1..10] of data_brg;
  i,j,banyak,
  Jum1,jum2,jum3 : integer;
begin
 jum1 :=0;
 jum2 :=0;
 jum3 :=0;
 write('Banyak record Max 10 =');readln(banyak);
  for i:= 1 to banyak do
   begin
    with barang[i] do
   begin
    writeln('Record ke -',i);
    write('Nama barang =');readln(namabrg);
     for j:= 1 to 3 do
       begin
         write('Unit barang ke- ',j,'= ');readln(unitbrg[j]);
        end;
     end;
   end;
    clrscr;
    writeln('---------------------------------');
    writeln('Nama barang  unit 1  unit2  unit3');
    writeln('---------------------------------');
    { cetak data }
    for i:= 1 to banyak do
     begin
   with barang[i] do
    begin
     jum1:=jum1+unitbrg[1];
     jum2:=jum2+unitbrg[2];
     jum3:=jum3+unitbrg[3];
     writeln(namabrg:15,unitbrg[1]:5,unitbrg[2]:5,unitbrg[3]:5);
    end;
   end;
    writeln('---------------------------------');
    writeln('Jumlah unit 1 =',jum1:6);
    writeln('Jumlah unit 2 =',jum2:6);
```

```
    writeln('Jumlah unit 3 =',jum3:6);
    readln;
end.
```

V. Tipe data record dengan field tipe record

Dalam Turbo Pascal tipe data record dapat didefinisikan juga sebagai field dari suatu record. Artinya suatu record dapat juga mempunyai field yang merupakan record. Contoh: sebuah data pegawai mempunyai struktur sebagai berikut :

- Nama pegawai -> string
- Mulai masuk -> - Tgl
  - Bln
  - Thn
- Alamat pegawai -> - Jalan
  - Kota
- Gaji -> - Gaji pokok
  - Lembur
  - Tunjangan

Maka dapat disusun program sebagai berikut :

Contoh :

```
program penggunaan_field_tipe_record;
uses crt;
 type
  masuk   = record
          tgl : 1..31;
          bln : 1..12;
          thn : integer;
         end;
  alamat  = record
          jalan : string[20];
          kota  : string[10];
         end;
  gajipeg = record
          pokok,tunjangan,lembur : real;
         end;

  datapegawai = record
              nama     : string[20];
              tglmasuk : masuk;
              almt     : alamat;
              gaji     : gajipeg;
             end;
  var
   pegawai    : array [1..10] of datapegawai;
   i,p,banyak : integer;
```

```
begin
clrscr;
  write('Banyak data record =');readln(banyak);
  for i := 1 to banyak do
    begin
     writeln('record ke -',i);
      with pegawai[i] do
       begin
        write('nama pegawai :');readln(nama);
        write('Tanggal masuk:');readln(tglmasuk.tgl);
        write('Bulan Masuk  :');readln(tglmasuk.bln);
        write('Tahun masuk  :');readln(tglmasuk.thn);
        write('Alamat       :');readln(almt.jalan);
        write('Kota         :');readln(almt.kota);
        write('Gaji pokok   :');readln(gaji.pokok);
        write('Tunjangan    :');readln(gaji.tunjangan);
        write('Lembur       :');readln(gaji.lembur);
       end;
    end;

    { cetak data }
   for i := 1 to banyak do
    begin
     writeln('record ke -',i);
      with pegawai[i] do
       begin
         writeln('nama       :',nama);
         writeln('Tanggal masuk:',tglmasuk.tgl);
         writeln('Bulan Masuk  :',tglmasuk.bln);
         writeln('Tahun masuk  :',tglmasuk.thn);
         writeln('Alamat       :',almt.jalan);
         writeln('Kota         :',almt.kota);
         writeln('Gaji pokok   :',gaji.pokok);
         writeln('Tunjangan    :',gaji.tunjangan);
         writeln('Lembur       :',gaji.lembur);
       end;
    end;
   readln;
   end.
```

VI. Record bervariasi

Record yang telah dibahas sebelumnya merupakan struktur record yang pasti, artinya field-field di dalam record sudah tertentu dan pasti. Selain itu di program Pascal dapat juga dibuat suatu record yang mempunyai field yang tidak pasti atau dapat berubah, yang disebut sebagai record yang bervariasi. Dalam record yang bervariasi dapat

mengandung suatu field yang tergantung dari suatu kondisi. Dalam penerapanya dalam program hanya dapat diterima satu buah field yang bervariasi saja. Field bervariasi ini harus terletak dibawah field yang tetap.

Contoh :

Ada sebuah struktur data pegawai yang terdiri dari :

- Nama pegawai
- Alamat pegawai
- Umur
- Gaji -> untuk gaji dibedakan antara pegawai tetap dgn honorer
  Untuk tetap - Tunjangan
  - Lembur
  - Gaji pokok
  Untuk honorer - Gaji pokok

Deklarasi dan program :

```
type
 Status_pegawai = (Honorer,Tetap);
 data_pegawai   = record
                  nama    : string[15];
                  alamat  : string[20];
                  umur    : byte;
                  case status : status_pegawai of
                  honorer : (gaji_h    : real);
                  tetap   : (gaji_t    : real;
                                tunjangan : real;
                                lembur    : real);
                 end;
```

Contoh :

```
type
 Status_pegawai = (Honorer,Tetap);
 data_pegawai   = record
                  nama    : string[15];
                  alamat  : string[20];
                  umur    : byte;
                  case status : status_pegawai of
                  honorer : (gaji_h    : real);
                  tetap   : (gaji_t    : real;
                                tunjangan : real;
                                lembur    : real);
                 end;
 var
  pegawai  : array[1..10] of data_pegawai;
  banyak,i : integer;
```

```
   kode     : char;
  begin
   { input data}
    write('Banyak Data max 10 :');readln(banyak);
    for i := 1 to banyak do
     begin
      with pegawai[i] do
     begin
      write('Nama pegawai =');readln(nama);
      write('Alamat       =');readln(alamat);
      write('Umur         =');readln(umur);
      write('pegawai tetap(T) atau Honorer(H) ');readln(kode);
       kode := upcase(kode);
       case kode of
        'H' : begin
                 Status:=Honorer;
                 write('Gaji didapat= ');readln(gaji_h);
                end;
        'T' : begin
                 status:= Tetap;
                 write('gaji tetap = ');readln(gaji_t);
                 write('Tunjangan  = ');readln(tunjangan);
                 write('Lembur     = ');readln(lembur);
                end;
         end;
       end;
     end;
      { cetak data}
writeln('-----------------------------------------------------');
writeln(' Nama Alamat  Umur  Status  Gaji  Tunjangan   Lembur');
writeln('-----------------------------------------------------');
    for i:= 1 to banyak do
     begin
    with pegawai[i] do
     begin
       write(nama:15);
       write(alamat:20);
       write(umur:3,' ');
      case status of
        honorer : writeln('Honorer',Gaji_h:8:2);
        Tetap   :
    writeln('Tetap  ',Gaji_t:8:2,tunjangan:8:2,lembur:8:2);
       end;
     end;
    end;
```

```
   writeln('-------------------------------------------');
   readln;
    end.

program contoh_record_bervariasi;
uses crt;
 type
  status_karyawan = (lajang,menikah,cerai);
  data_karyawan = record
                nama    : string[15];
                alamat  : string[20];
                 gaji      : integer;
                case status : status_karyawan of
                 lajang  :();
                 menikah :(anakm : 0..20);
                 cerai   :(anakc : 0..20; lagi :char);
                end;
  var
   karyawan : array [1..10] of data_karyawan;
   i,banyak : integer;
   sts : char;
  begin
   clrscr;
   write('Jumlah data record :');readln(banyak);
    for i := 1 to banyak do
     begin
      with karyawan[i] do
       begin
         write('Nama   =');readln(nama);
         write('Alamat =');readln(alamat);
         write('Gaji   =');readln(gaji);
  write('status M=menikah L=lajang C=cerai');readln(sts);
   if upcase(sts)='L' then
    begin
      status:=lajang;
    end;
   if upcase(sts)='M' then
    begin
      status:=Menikah;
      write('Jumlah anak= ');readln(anakm);
    end;
   if upcase(sts)='C' then
    begin
      status:=Cerai;
      write('Jumlah anak      = ');readln(anakc);
```

```
      write('Kawin lagi (Y/T) = ');readln(lagi);
    end;
  end;
 end;
{ tampil}
    for i := 1 to banyak do
     begin
      with karyawan[i] do
       begin
         write(nama);
         write(alamat);
         write(gaji);
   case status of
    lajang   : writeln('lajang');
    menikah  : begin
               writeln('menikah','  ',anakm:4);
               end;
    cerai      : begin
                 writeln('cerai ', '  ',anakc:4,' ',lagi);
                 end;
    end;
end;
end;
readln;
end.
program penggunaan_field_tipe_record;
uses crt;
 type
  masuk   = record
          tgl : 1..31;
          bln : 1..12;
          thn : integer;
         end;
  alamat  = record
           jalan : string[20];
           kota  : string[10];
          end;
  gajipeg = record
           pokok,tunjangan,lembur : real;
          end;
  datapegawai = record
              nama     : string[20];
              tglmasuk : masuk;
              almt     : alamat;
              gaji     : gajipeg;
```

```
            end;
  var
   pegawai    : array [1..10] of datapegawai;
   i,p,banyak : integer;
begin
clrscr;
  write('Banyak data record =');readln(banyak);
   for i := 1 to banyak do
     begin
      writeln('record ke -',i);
       with pegawai[i] do
        begin
         write('nama pegawai :');readln(nama);
         write('Tanggal masuk:');readln(tglmasuk.tgl);
         write('Bulan Masuk  :');readln(tglmasuk.bln);
         write('Tahun masuk  :');readln(tglmasuk.thn);
         write('Alamat       :');readln(almt.jalan);
         write('Kota         :');readln(almt.kota);
         write('Gaji pokok   :');readln(gaji.pokok);
         write('Tunjangan    :');readln(gaji.tunjangan);
         write('Lembur       :');readln(gaji.lembur);
        end;
     end;
   { cetak data }
   for i := 1 to banyak do
     begin
      writeln('record ke -',i);
       with pegawai[i] do
        begin
         writeln('nama       :',nama);
         writeln('Tanggal masuk:',tglmasuk.tgl);
         writeln('Bulan Masuk  :',tglmasuk.bln);
         writeln('Tahun masuk  :',tglmasuk.thn);
         writeln('Alamat       :',almt.jalan);
         writeln('Kota         :',almt.kota);
         writeln('Gaji pokok   :',gaji.pokok);
         writeln('Tunjangan    :',gaji.tunjangan);
         writeln('Lembur       :',gaji.lembur);
        end;
     end;
   readln;
   end.
```

Procedure

Procedure adalah suatu program yang terpisah dalam block tersendiri yang berfungsi sebagai subprogram (program bagian). Penggunaan prosedur diawali dengan kata cadangan procedure di dalam bagian deklarasi procedure. Pemanggilan procedure dengan menggunakan judul procedure.

Pada program terstruktur banyak menggunakan procedure karena :

- Sebagai penerapan program yang modular yaitu memecah program yang rumit menjadi program- program bagian yang lebih sederhana dalam bentuk procedure.
- Untuk beberapa perintah yang sering digunakan berulang, cukup dituliskan sekali dalam procedure dan dapat dipanggil sewaktu-waktu.

Contoh Procedure tanpa parameter,

```
Procedure garis;
begin
 writeln('----------');
end;
procedure Judul;
begin
 writeln('pascal');
end;
{modul utama}
begin
  garis;
  judul;
  garis;
end.
```

hasil :

```
----------
pascal
----------
```

I. Parameter dalam procedure

Nilai di dalam suatu procedure sifatnya adalah local, berarti hanya dapat digunakan oleh procedure tersebut saja dan tidak dapat digunakan oleh procedure yang lain.

Contoh :

```
procedure hitung;
var
  a,b,c : integer;
  begin
   write('Nilai a =');readln(a);
   write('Nilai b =');readln(b);
   c:=a+b;
    writeln('hasilpenjumlahan=',c:5);
    readln;
```

```
 end;
{ modul utama }    akan salah jika pada modul utama :
begin                  begin
  hitung;                hitung;
end.               writeln('nilai a=',a); -> a tdk dikenal
                         end.
```

Pada kasus diatas dapat diselesaikan dengan menggunakan deklarasi secara global, sehingga semua procedure dibawah deklarasi global dapat menggunakannya.

Contoh penggunaan deklarasi global :

```
uses crt;
procedure kali;
var
  a,b,c : integer;  { deklarasi secara local utk proc. kali saja}
  begin
    write('A =');readln(a);
    write('b =');readln(b);
    c:=a*b;
    writeln('hasil c =',c:5);
  end;
var
  d,e,f  : integer;    {deklarasi secara global  hingga  dikenal
oleh}
procedure jumlah; {proc.jumlah&procedure dibawahnya  }
    begin
      write('nilai d =');readln(d);
      write('nilai e =');readln(e);
      f:=d+e;
      writeln('nilai f =',f:5);
    end;
procedure kurang;   {procedure ini menggunakan varibel global}
     begin     { yang terletak diatas procedure jumlah}
   write('Nilai d =');readln(d);
   write('nilai  e =');readln(e);
   f:= d-e;
   writeln('Nilai f=',f:5);
    end;
 { modul utama}
   begin
    clrscr;
    kali;
    jumlah;
    kurang;
    readln
   end.
```

II. Pengiriman parameter secara Nilai

Pada pengiriman parameter secara nilai (by value), parameter formal akan berisi nilai yang dikirimkan dari parameter nyata dan nilai parameter tersebut akan local diprocedure yang dikirim. sifat dari pengiriman nilai ini adalah satu arah, sehingga perubahan nilai dari parameter formal tidak akan mempengaruhi nilai parameter nyata.

Contoh :

```
Program pengiriman_parameter_secara_nilai;
procedure kali(a,b : integer); {parameter formal}
 var
  hasil : integer;   {local variabel}
 begin
  hasil :=a*b;
  writeln('hasil =',hasil:6);
 end;
{modul Utama}
 var
  bil1,bil2 : integer;
 begin
  write('bilangan 1 =');readln(bil1);
  write('bilangan 2 =');readln(bil2);
  kali(bil1,bil2); {parameter nyata}
  readln;
 end.
```

Di bawah ini merupakan contoh bahwa perubahan pada parameter formal tidak akan mempengaruhi nilai parameter nyata, karena sifatnya adalah satu arah.

```
Procedure kali(a,b : integer);
        kali(bil1,bil2);
```

Contoh:

```
Program parameter_nilai_tdk_mempengaruhi_parameter_nyata;
uses crt;
procedure test_hitung(a,b,hasil : integer);
 begin
  hasil := a*b;
  writeln('A =',a:4,' B=',b:4,' Hasil=',hasil:6);
 end;
{modul utama}
 var
  bil1,bil2,bil3 : integer;
 begin
  bil1:=3;bil2:=4;bil3:=0;
  test_hitung(bil1,bil2,bil3);
  writeln('bil1=',bil1:4,' bil2=',bil2:4,' bil3=',bil3);
  readln;
 end.
```

III. Pengiriman parameter secara acuan (by reference)

Sifat dari pengiriman parameter secara acuan adalah dua arah artinya perubahan dari parameter formal akan mempengaruhi nilai dari parameter nyata. Cara deklarasi di procedure dengan kata cadangan Var seperti berikut :

```
procedure kali(Var a,b,c : integer);  -> parameter formal
          kali(x,y,z);              -> parameter nyata
```

Contoh :

```
program pengiriman_parameter_secara acuan;
uses crt;
procedure kali(var a,b,c : integer); {parameter formal  acuan}
begin
  c:=a*b;
end;
{modul utama}
var
  x,y,z : integer;
begin
  write('nilai x=');readln(x);
  write('nilai y=');readln(y);
  kali(x,y,z);  {mengirimkan parameter secara acuan}
  writeln('Nilai z =',z:5);
end.
```

Contoh penggunaan parameter secara acuan untuk perhitungan faktorial:

```
program Contoh_penggunaan_parameter_acuan;
uses crt;
 procedure faktor(var banyak,hasil : integer);
  var
   i : integer;
  begin
    hasil := 1;
     for i := 1 to banyak do
      begin
    hasil := hasil*I;
      end;
  end;
{modul utama}
  var
    n,jumlah : integer;
  begin
    write('Berapa faktorial =');readln(n);
    faktor(n,jumlah);
    writeln(n:5,' faktorial adalah =',jumlah:6);
```

```
  readln;
 end.
```

Contoh Program dengan penggunaan procedure dgn parameter secara acuan pada perhitungan pangkat lebih besar dari 2 :

```
program pangkat;
uses crt;
 procedure pangkat(var bil,hasil:real;pang:integer);
  var
  i : integer;
begin
  hasil :=1;
    for i:= 1 to pang do
      begin
    hasil:=hasil*bil;
      end;
    end;
{modul utama}
 var
    angka,hasil : real;
    pang : integer;
 begin
  clrscr;
  write('bilangan yang dipangkat =');readln(angka);
  write('dipangkatkan =');readln(pang);
  pangkat(angka,hasil,pang);
  write('hasil =',hasil:5:2);
  readln;
 end.
```

IV. Procedure memanggil procedure yang lain

Di dalam pascal diperkenankan procedure memanggil procedure yang lain seperti contoh berikut :

```
program procedure_memanggil_procedure_yang_lain;
uses crt;
 procedure satu(a1: integer);
   begin
    writeln(' nilai a =',a1:2,' ada diprocedure satu');
   end;
 procedure dua(a2: integer);
   begin
    writeln(' nilai a =',a2:2,' ada diprocedure dua');
    satu(a2);
   end;
 procedure tiga(a3: integer);
```

```
  begin
   writeln(' nilai a =',a3:2,' ada diprocedure tiga');
   dua(a3);
  end;
procedure empat(a4: integer);
  begin
   writeln(' nilai a =',a4:2,' ada diprocedure empat');
   tiga(a4)
  end;
  {modul Utama}
   var
    a : integer;
   begin
    clrscr;
    write('nilai a=');readln(a);
    empat(a);
    readln;
   end.
```

V. Procedure Tersarang

Procedure tersarang adalah procedure yang terdapat di dalam procedure yang lain dan dipanggil oleh procedure diluarnya.

```
program contoh_procedure_tersarang;
uses crt;
procedure satu;          {deklarasi procedure satu}
 procedure dua;
  begin                  {awal procedure dua}
   writeln('procedure dua ada di procedure satu');
   end;                  {akhir procedure dua}
 procedure tiga;
  begin                  {awal procedure tiga}
   writeln('procedure tiga ada di procedure satu');
  end;                   {akhir procedure tiga}
begin                    {awal procedure satu}
 writeln(' procedure satu');
 dua;                    {memanggil procedure dua}
 tiga;                   {memanggil procedure tiga}
end;                     {akhir procedure satu}
{modul utama}
begin
 clrscr;
 writeln(' modul utama');
 satu;                   {memanggil procedure satu}
 readln;
end.
```

VI. Procedure memanggil dirinya sendiri (rekursi)

Di dalam pascal diperkenankan memanggil procedurenya sendiri. istilah ini disebut sebagai recursion. Dalam penggunaanya membutuhkan memory yang besar. Karena pada setiap pemanggilan sejumlah memory tambahan dibutuhkan.

Contoh :

```
program procedure_memanggil_dirinya_sendiri;
uses crt;
var
 I : integer;
 procedure rekursi;
  begin
    writeln('pemangilan procedure ke-',i:5);
    i:=i+1;
    if i < 5 then rekursi;
  end;
 {modul utama}
  begin
    clrscr;
    i:=1;
    rekursi;
    readln;
  end.
```

Function

Blok pada function hampir sama dengan blok pada procedure, hanya pada function harus dideklarasikan dengan tipe dari function tersebut yang merupakan tipe hasil dari function itu sendiri. Sehingga dikatakan function dapat mengembalikan nilai.

Sintaks :

```
FUNCTION identifier(daftar parameter) : type;
```

I. Parameter Nilai dalam function

Parameter dalam function dapat dikirimkan secara nilai atau secara acuan. Penulisan judul function yang menggunakan parameter secara Nilai adalah :

```
Function besar(a,b  : real) : real;
```

Contoh :

```
program penggunaan_parameter_nilai;
uses crt;
function besar(a,b :real) : real;
 begin
  if a>b then
     besar:=a
     else
     besar:=b;
 end;
{modul utama}
 var
 nil1,nil2 : real;
 begin
  write('bilangan 1=');readln(nil1);
  write('bilangan 2=');readln(nil2);
  writeln('bilangan terbesar =',besar(nil1,nil2):6:2);
  readln;
 end.
```

II. Function dengan parameter acuan

Penulisan judul function dengan menggunakan parameter secara acuan adalah sama dengan procedure yaitu ditambah Var pada deklarasi parameter. Dengan demikian nilai parameter acuan ini dapat digunakan sebagai hasil balik.

Sintaks :

```
FUNCTION  jumlah(var a,b : integer) : integer;
```

Contoh :

```
program pengiriman_parameter_secara_acuan;
function kali(var bil1,bil2,jumlah : integer) : integer;
 begin
```

```
  kali:=bil1*bil2;
  jumlah:=bil1+bil2;
 end;
 var
x,y,z : integer;
  begin
   write('bilangan 1=');readln(x);
   write('bilangan 2=');readln(y);
   writeln(x:3,'*',y:3,' = ',kali(x,y,z):5);
   writeln(x:3,'+',y:3,' = ',z);
   readln;
 end.
```

III. Function tanpa parameter

Suatu function tanpa parameter berarti nilai balik yang akan dihasilkan merupakan nilai yang sudah pasti. Jika pada function dengan parameter, parameternya digunakan untuk input pada function dan function akan memberikan hasil balik sesuai dengan parameter yang diberikan sehingga bisa diatur dari program pemanggil. Sedang pada function tanpa parameter hasil dari function tidak dapat diatur. Sehingga function tanpa parameter jarang digunakan.

Contoh :

```
function tiga : integer;
 begin
  tiga:=3;
 end;
begin
 writeln(tiga);
end;
```

Jadi hasil :

3

Function type string untuk membuat garis, ini juga merupakan contoh function tanpa parameter.

```
uses crt;
  function garis : string;
   begin
    garis:='----------';
   end;
{modul utama}
 begin
  writeln(garis);
  writeln('pascal');
  writeln(garis);
```

```
 readln;
end.
```

Contoh :

```
program pangkat_dgn_function;
uses crt;
 function pangkat(bil :real; pang: integer) : real;
var
   hasil : real;
   i     : integer;
  begin
   hasil := 1;
   for i:= 1 to pang do
    begin
     hasil:= hasil*bil;
    end;
     pangkat:=hasil;
  end;
  var
   hitung,bil : real;
   pang : integer;
  begin
    write('bilangan =');readln(bil);
    write('pangkat=');readln(pang);
    hitung:= 2*pangkat(2,3);
writeln(bil:5:2,' pangkat',pang:5,' = ',pangkat(bil,pang):6:2);
    writeln('2 * (2 pangkat 3) =',hitung:6:2);
readln;
  end.
```

IV. Rekursi pada function

Rekursi adalah dimana suatu function memanggil dirinya sendiri. Proses dapat dilihat pada contoh berikut. Dimana fungsi faktor dipanggil oleh dirinya sendiri.

Contoh :

```
program function_memanggil_funnction_yg_lain;
uses crt;
function faktor(bilangan : integer) : real;
 begin
  if bilangan=0 then
     faktor:=1
   else
     faktor:=faktor(bilangan-1)*bilangan;
  end;
  var
   n : integer;
```

```
begin
 write('berapa faktorial =');readln(n);
 writeln(N:5,' faktorial =',faktor(n):9:0);
 readln;
end.
```

File Teks

Pascal mempunyai dua macam file. File teks dan file binary. Bagian ini membicarakan file teks. Fiel teks tidak mempunyai besar yang eteap. Untuk menandai akhir suatu file, komputer menempatkan karakter khusus end-of-file (<eof>) setelah karakter yang paling akhir. Untuk menandai akhir suatu baris, komputer menampatkan karakter khusus end-of-line pada akhir baris.

Dalam perogram yang interaktif biasanya kita menuliskan nilai sentinel untuk menandai akhir suatu baris atau file. Sebagai contoh, pecahan program semacam ini digunakan membaca suatu nama (kumpulan karakter) dengan nilai sentinel titik.

```
Read(nama);
While(nama<>'.') Do
        Read(nama);
```

Dalam file teks, untuk mengetes apakah baris sudah berganti, kita bias menggunakan fungsi eoln. Berikut ini pecahan program diatas yang ditulis deng fungsi eoln.

```
While not EoLn Do
        Read(nama);
```

Untuk mengetes apakah akhir suatu file, kita bias menggunakan fungsi eof seperti berikut ini.

```
While not Eof(InfileData)  Do
        Begin
                While Not EoL Do
                        Read(Nama);
                ReadLn(Gaji);
        End;
```

InfileData diatas merupakan nama file yang bertipe teks. Program diatas membaca variable-variabel Nama dan gaji dalam File Infile. Apabila akhir baris dtidak ditemui, nilai Eoln berarti false yang berarti program membaca variable nama. Setelah akhir baris ditemui, nilai EoLn menjadi true dan program membaca variable berikutnya yaitu Gaji. Setelah akhir file ditemui, nilai Eof menjadi true dan program keluar dari loop.

Membuat File Teks

File teks bias dibuat melalui beberapa cara. Kalau kita ad DOS, maka dengan cara yang termudah adalah dengan menggunakan DOS tersebut. Perintah yang digunakan adalah sebagai berikut :

```
A:\> Edit <namafile>
```

Dos editor kemudian muncul, dan kita bias mengetik angka-angka atau huruf yang akan disimpan sebagai file teks.

Apabila kita mempunyai pascal Editor, file teks dapat dibuat dengan menggunakan editor pascal. Bentuk editor tersebut mirip dengan DOS Editor. Untuk mengaktifkan menu, kita bisa menekan tombol F10. File teks bisa disimpan dengan menu Save atau Save As.

Deklarasi File Teks

Seperti variable-variabel lain dalam pascal, file teks juga harus dideklarasikan terlebih dahulu sebelum digunakan. Berikut ini adalah deklarasi file teks bernama InfileData.

```
Program ProsesFile (InfileData,OutFIle)
Var
        InfileData, Outfile      : text;
```

InfileData dan OutFile dideklarasikan sebagi file teks. Dalam judul program keduanya harus dituliskan. Apabila program juga akan menggunakan keyboard (sebagai input) dan monitor (sebagai Output), maka judul program dituliskan sebagai berikut :

```
Program ProsesFile (InfileData,Input,OutFile,Output);
Var
        InfileData,Outfile       : Text;
```

Pernyataan Reset

Pernyataan Reset digunakan untuk menyiapkan suatu file teks untuk dibaca oleh program. File teks siap untuk diproses dengan pernyataan berikut :

```
Reset(InfileData);
```

Dengan pernyataan Reset, pointer digeser ke permulaan file Teks. Karakter pertama dalam suatu fiel akan diproses sesudah pernyataan Reset. Sebelum data dibaca, operasi Reset harus dilakukan , apabila tidak program akan gagal menjalankan tugasnya (error akan muncul).

Pernyataan Rewrite

Untuk menyiapkan Output (file teks yang akan menampung Output program kita), kita harus menuliskan pernyataan seperti berikut ini :

```
Rewrite(OutFile);
```

Pernyataan diatas menyiapkan file OutFile untuk menampung hasil pemrosesan. Kija tidak ada file OutFile sebelumnya, OutFile akan diciptakan. Apabila
sebelumnya ada file OutFile, pointer akan ditempatkan pada awal File dan semua isi OutFile yang lama akan terhapus oleh hasil pemrosesan yang terbaru.

Pernyataan Close

Pernyataan Close dipakai untuk menutup file-file yang dibuka dan dipakai dalam suatu program. Program yang menggunakan operasi Output-Input (O/I) biasanya lebih lambat, karena program tersebut dengan menggunakan jasa DOS berhubungan dengan aspek Fisik dari disket.

Pascal menyediakan memori untuk menampung atau menuliskan data ke file. Ketika program menuliskan data atau membaca data, program membaca atau menuliskan data ke file buffer., bukannya langsung ke file eksternal secara langsung. Pascal secara periodic memindahkan data tersebut dari file buffer ke file eksternal. Apabila kita tidak menuliskan pernyataan close, proses pemindahan data tidak akan sempurna, dengan

akibat ada data yang hilang. Tidak disimpan dalam file. Dengan cara semacam itu, program yang melibatkan operasi I/O akan diproses lebih cepat daripada apabila program langsung memanggil file eksternal.
Penulisan pernyataan Close adalah sebagai berikut :

```
Close(InfileData);
```

Contoh program :

```
Program hasilPrinter (InfileData,Output):
Var
        InfileData      : text;
        I               : Integer;
Begin
AssiGn(InfielData,'PRN');
Rewrite(InfileData)
WriteLn(InfileData,'Bilangan dari 10 ke 10);
WriteLn;
For     I := 1 to 10 Do
        WriteLn(InfileData,I);
        WriteLn(InfileData,chr(12));
Close(InfileData);
End.
```

Hasil dari program diatas :

```
1
2
3
4

5
6
7
8
9
10
```

File Binary

Sebagai alternatif penulisan file teks, pascal memungkinkan kita menuliskan file dengan menggunakan kata(konstruktor) file seperti berikut ini :

```
Type
    DeretAngka = File of Integer;
Var
    IntData     : DeretAngka;
    Angka       : Integer;
```

File inData disebut juga sebagai File Binary. File Binary adalah file dimana representasi internal dari tiap-tiap komponen secara langsung. Misalkan nilai variable angka adalah 244, pernyataan :

```
    Write(InData,Angka);
```

Mengkopi representasi binary internal variable angka dari memori ke file InData. Misalkan file OutData bertipe teks, pernyataan berikut ini :

```
    Write(OutData, angka:4);
```

Akan menuliskan niali variable angka ke file OutData dengan empat Bytes. Komputer pertama harus mengubah representasi binary dari memori ke string '244' dan kemudian menuliskan kode bnary unutk karakter blank(' ') ,2,4 dan 4 ke OutFile.

Sebaliknya apabila angka '244' mau ditampilkan dilayar monitor, komputer akan mengkopi representasi binary dair blank (' '),2,4 dan 4 kemudian menuliskan ke teks string '244' yang kemudian ditampilkan dilayar monitor. Proses semacam ini memakan waktu lebih lama dibandingkan kalau langsung mengkopi representasi binary internal ke disk.

Bentuk Umum dari file binary(sering juga disebut typed filed), adalah sebagai berikut :

```
Var
    InFile : File of <tipe>;
```

Dimana tipe bis merupakan tipe dasar file seperi Integer, Char, bahkan suatu record, dan bias juga suatu string.

Berikut deklarasi file binary :

```
Type
    String10    = string[10];
    RecMhs      = record
            Nama  : string[10];
            IP    : Real;
    End;


Var
    InChar      : File of char;
    InMhs       : File of recMhs;
    InItgr      : File of Integer;
    InStrng     : File of String[10];
```

Contoh program :

```
Program Bin01(input,OutFile);
Var
        OutFile                 : File of Integer;
        Angka,Jumlah            : Integer;
Begin
AssiGn(OutFile,'a:\latihan\outline.txt');
Rewrite(OutFile);
WriteLn;
WriteLn('Berapa angka yang akan dimasukkan  : ');ReadLn(Jumlah);
For Angka := 1 to Jumlah Do
        Write(OutFile,Angka);
Reset(OutFIle);
For Angka := 1 to Jumlah Do
        Begin
                Read(OutFile, Angka);
                Write(Output,Angka);
        End;
ReadLn;
End.
```

Pointer

Pada bab terdahulu (dalam hal Array), masih menyisakan beberapa kelemahan, antara lain adalah penggunaan space (ruang) yang tetap. Kita harus tahu jumlah maksimum yang akan kita alokasikan untuk record tersebut, baru kita menentukan space yang kita butuhkan.

Pascal menyediakan fasilitas untuk mengatasi masalh static data structure seperti digambarkan diatas dengan menggunakan dynamic data structure. Berbeda dengan static data structure, dynamic data structure, struktur data bias berkembang atau berkurang sesuai dengan kebutuhan. Dengan dynamic data structure kita tidak perlu menentukan jumlah record maksimum dan tidak perlu membuang space yang tidak terpakai apabila jumlah record lebih kecil dibandingkan kapasitas yang disediakan.

Sebagai gambaran array record dan linked record, adalah sebagai berikut :

Misalkan kita mempunyai data yang tersusun berdasarkan urutan alphabet adalah sebagai berikut :

Pertama (6)		Kedua (7)	
Nama	(1) Djoko	5.000	5
Nama	(2) Koen	7.000	-1
Nama	(3) Laudiah	3.000	4
Nama	(4) Boby	4.000	1
Nama	(5) Inonk	6.000	2
Nama	(6) Inem	8.000	3
Nama	(7) ….	….	….
Nama	(8) ….	….	….
.			
.			
.			
.			
Nama	(50) ..	…	50

Data diatas bisa dibaca sebagia berikut :

Pertama menunjuk ke posisi (6) yang merupakan urutan pertama. Kemudian Inem menunjuk kearah berkutnya (Laudiah) yang terletak pada baris ketiga. Laudiah menunjuk keposisi berikutnya daalam urutan yaitu Boby yang berada pada urutan ke-4. Posisi terakhir, yaitu Koen menunjuk pada posisi -1 yang berarti daftar sudah berakhir.

Data diatas dapat dideklarasikan sebagai berikut :

```
Type
        Pembeli   : Record
                Nama          : Packed ArraY [1..10] of char;
                Uang, Link    : IntegeR;
        End;
Var
        Daftar                : ArraY [1..50] of Pembeli;
        Pertama, kosong       : Integer;
```

Apabila kita ingin menambah data baru, maka kita bis menyelipkan data tersebut diantara daftar yang ada sekarang. Misalkan kita ingin memasukkan nama baru mahmud kedalam data diatas.

Data Tipe Pointer

Kita akan membicarakan data dengan tipe pointer. Data bisa kita simpan pada variable semacam ini. Sebagai contoh, dibawah ini penulisan deklarasi variable dengan tipe pointer.

```
Type
        JumlahReal   = ^Real;
Var
        Jumlah     : JumlahReal;
```

Jumlah dideklarasikan sebagai variable pointer dengan tipe JumlahReal. Kita bisa menyimpan alamat memori variable tipe real pada jumlah. ^Real dibaca sebagi petunjuk (pointer) ke Real.

Dengan Pointer, record daftar pembeli seperti dimuka bisa ditulis sebagi berikut :

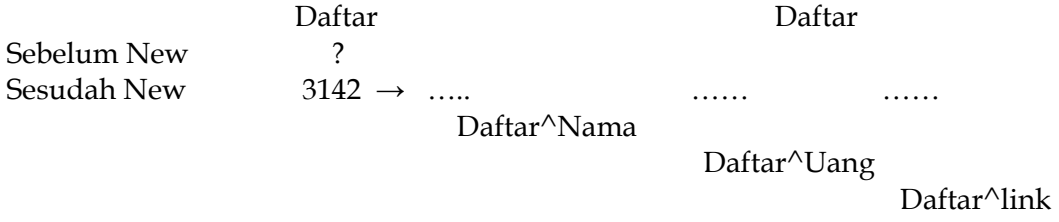
```
Type
        Pembeli  = Record
                Nama          : Packed ArraY [1..10] of char;
                Uang          : Integer;
                Link          : ^Pembeli;
        End;
Var
Daftar, Daftar1,daftar2 : ^Pembeli;
```

Perlu diingat bahwa variable pointer ini hanya mengandung nilai yang menunjuk pada alamat memori yang menunjuk pada record dengan tipe pembeli.

Pernyataan New digunakan untuk mengaktifkan daftar.

```
New(daftar);
```

Pernyataan tersebut mengalokasikan pada daftar. Alamat memori kemudian disimpan pada variable daftar. Gambar berikut menjelaskan proses tersebut secara grafis.



Dengan pernyataan New, sel memory dengan alamat 3142 dialokasikan ke daftat. Penugasan semacam dibawah ini akan mengisi nilai untuk daftar^Nama dan Daftar^Uang. Daftar^link belum diisi nilainya.

```
        Daftar^nama  : Sari;
```

Daftar^uang : 10000;

Kita juga bisa mengkopikan isi daftar ke daftar1 yang mempunyai tipe data yang sama.

Daftar1 := Daftar;

Daftar dengan daftar1 mempunyai alamat memory yang sama, dan dengan demikian menunjuk pada isi yang sama. Gambar berikut ini menjelaskan proses tersebut secara grafis.

	Nama	Uang	Link
Daftar	Sari	10000	?

Sesudah daftar dikopikan kedalam daftar1, adalah sebagai berikut :

	Nama	Uang	Link
Daftar	Sari	10000	?

Daftar1 ——————————————————————————————↑

Menghubungkan variable pointer

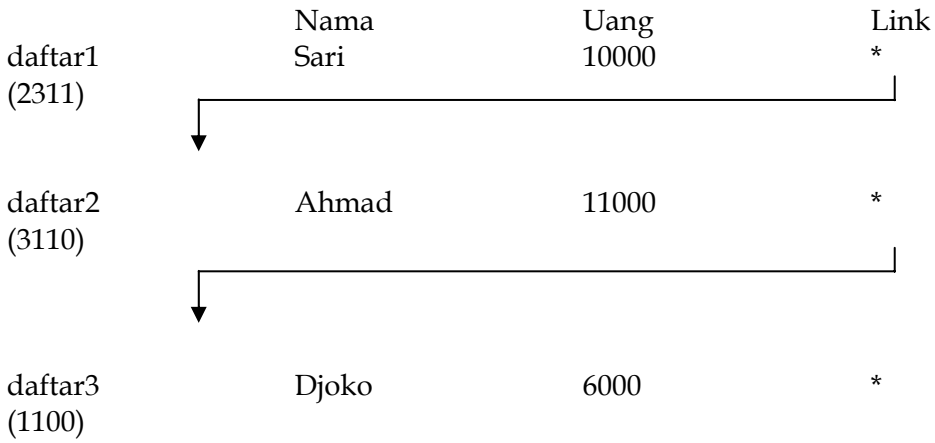
Misalkan kita mempunyai 3 variabel pointer (sering juga disebut Node), seperti yang ada dibawah ini :

	Nama	Uang	Link
Daftar1 (2311)	Sari	10000	?
Daftar2 (3110)	Ahmad	11000	?
Daftar3 (1100)	Djoko	6000	?

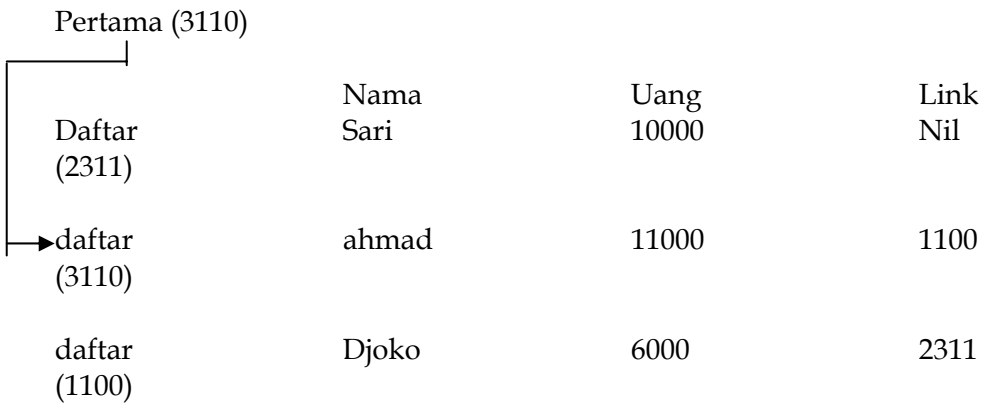
Angka dalam kurung adalah menunjuk alamat memori variable diatas. Variabel diatas belum dihubunkan satu dengan yang lain. VAriabel diatas dihubungkan dengan pernyataan sebagai berikut :

1. Daftar1^link := daftar2;
2. Daftar2^link := daftar3;
3. Daftar3^link := Nil;

sesudah penugasan pernyataan-pernyataan diatas, variable diatas bisa digambarkan sebagai berikut :



Nil menunjukkan bahwa daftar3 merupakan akhir dari daftar pembeli diatas. Dengan menggunakan alamat-alamat memori, proses diatas akan nampak jelas :



variable pertama mempunyai alamat memori yang menunjuk pada alamat 3110.

Pertama^Nama = ahmad
Pertama^Uang = 11000
Pertama^Link = 1100

Pernyataan daftar := pertama^link (atau dalam hal ini daftar := 1100), membuat program bergerak turun ke memory sel 1100

Daftar^Nama = Djoko
Daftar^uang = 6000
Daftar^link = 2311

Pernyataan daftar := daftar^link (daftar := 2311) membuar program bergerak turun ke memory sel 2311

```
Daftar^Nama     = Sari
Daftar^Uang     = 10000
Daftar^Link     = Nil
```

Nil menunjukkan bahwa nilai ini merupakan tanda akhir list (daftar)
Contoh :
Berikut adalah program menciptakan linked data, mencari data dan menghilangkan data

```
Program Link1 (Input,Output);
Uses crt;
    Type
        Pointer     = ^Cell;
        Cell        = Record
            Value       : Integer;
            Next        : pointer;
        End;
Var
    Last, belakang, Q, P        : pointer;
    Angka, Nomor                : Integer;
    Jawab                       : Char;
    Found                       : Boolean;

Procedure printlist;
Begin
Last  := Belakang;
    While last <> nil Do
        Begin
            WriteLn(last^.Value);
            Last := Last^.next;
        End;
End;

Procedure Look;
Begin
    WriteLn('Masukkan Angka yang akan dicari  : ');ReadLn(angka);
    Last := Belakang;
    Found := false;
    While (last <> nil) and (found <> true) Do
        Begin
            If last^.value = angka then
                Begin
                    Found := True;
                    WriteLn('Angka Ditemukan');
                    WriteLn(Last^.value);
                    Found := True;
```

```
                    End;
                Else

                    Last := Last^.next;
                End;
End;

Procedure Delete;
Begin
WriteLn('Angka lain dihapus  ???');ReadLn(angka);
Last := Belakang;
Found := False;
While (last <> Nil) and (found <> true) Do
    Begin
        Last := Last^.next;
            If Q^.value     = angka then
                Begin
                    Dispose(Q);
                    Found := True;
                End;
            Else
            If last ^.value  = angka then
                Begin
                    WriteLn(last^.value, ' ' , ditemukan dan dihapus')l
                    Q^.next := last^.next;
                    Dispose(last);
                    Found := true;
                End;
            End;
End;

Begin
    Belakang := Nil;
    Nomor := 1;
    writeLn('nomor 1 : ', nomor);
    Jawab := 'y';
    While (jawab <> 't') Do
        Begin
            Writeln('Masukkan Angka   : ');ReadLn(angka);
            New(last);

            Last^.Value    := angka;
            Last^.next     := belakang;
            Belakang       := last;
            Nomor          := nomor + 1;
```

```
                    WriteLn('Nomor    : ', nomor);
                    WriteLn('Terus (y/t ???? ');
                    Readln(jawab);
            End;
printList;
Look;
Delete;
printList;
ReadLn(jawab);
End.
```

Hasil dari program diatas, adalah :

```
Masukkan angka       : 3
Terus(y/t)  ????  y

Masukkan angka       : 10
Terus(y/t)  ????  y

Masukkan angka       : 7
Terus(y/t)  ????  t

7
10
3

Masukkan angka yang akan dicari    : 10
Angka ditemukan
Angka ingin dihapus                : 7
7 ditemukan dan dihapus
10
3
```