A. 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.
```

B. 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.

```
   Daftar     Daftar
Sebelum New        ?
Sesudah New     3142      …..                    ……              ……
              Daftar^Nama
                                                                Daftar^Uang
          Daftar^link
```

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 :

	Nama	Uang	Link
daftar1 (2311)	Sari	10000	*
daftar2 (3110)	Ahmad	11000	*
daftar3 (1100)	Djoko	6000	*

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

Pertama (3110)

	Nama	Uang	Link
Daftar (2311)	Sari	10000	Nil
daftar (3110)	ahmad	11000	1100
daftar (1100)	Djoko	6000	2311

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
```