Catatan Kuliah

Pemrograman Komputer I

disusun oleh

Nanda Arista Rizki, M.Si.

PROGRAM STUDI MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS MULAWARMAN
2019

Daftar Isi

BAB 1

Pengenalan Bahasa Pascal

1.1 Perulangan *repeat..until*

Perintah perulangan proses dalam sebuah program bahasa Pascal adalah ***for..to..do***, ***while..do***, dan ***repeat..until***. Struktur untuk perintah ***repeat..until*** adalah sebagai berikut.

```
1 REPEAT
2 pernyataan_1;
3 pernyataan_2;
4 pernyataan_3;
5 ...
6 pernyataan_n;
7 UNTIL kondisi;
```

Perulangan ***repeat..until*** digunakan untuk melakukan kumpulan pernyataan sampai pada suatu kondisi tertentu. Perhatikan bahwa perulangan ini tanpa menggunakan blok ***BEGIN-END***. Perulangan ini akan mengeksekusi minimal sekali, walaupun kondisinya tidak terpenuhi. Hal ini karena sintaksnya dimulai dari kumpulan pernyataan dan diikuti oleh suatu kondisi.

Perulangan ***repeat..until*** dapat digunakan untuk menghitung nilai faktorial $n!$. Berikut adalah sintaks untuk menghitung nilai 6!.

```
...
faktorial:=1;
indeks_hitung:=6;
REPEAT
  faktorial:=faktorial*indeks_hitung;
  indeks_hitung:=indeks_hitung-1;
UNTIL indeks_hitung=0;
```

Berdasarkan sintaks tersebut, ketika peubah "indeks_hitung" bernilai 0, maka perulangan dihentikan. Sebagai informasi, jika ingin menghitung 8! maka ubahlah pernyataan untuk peubah "indeks_hitung" menjadi 8.

Sintaks untuk menghitung nilai faktorial menggunakan perulangan ***repeat..until*** dapat dimodifikasi. Perhatikan sintaks berikut.

```
...
faktorial:=1;
indeks_hitung:=1;
REPEAT
  faktorial:=faktorial*indeks_hitung;
  indeks_hitung:=indeks_hitung+1;
UNTIL indeks_hitung=6+1;
```

Berdasarkan sintaks tersebut, perulangan dihentikan ketika peubah "indeks_hitung" bernilai 7.

Selanjutnya, jika ingin membuat suatu program yang menggunakan perulangan ***repeat..until*** di dalam perulangan bersarang ***while..do***, maka perhatikan program berikut.

```
program contoh_repeat;
uses wincrt;
var
  faktorial:real;
  bilangan,indeks_hitung:integer;
begin
  writeln('berikan suatu bilangan');
  write('tekan 0 jika ingin keluar ');
```

```
9   readln(bilangan);
10  while bilangan <> 0 DO
11  begin
12    faktorial:=1;
13    indeks_hitung:=1;
14    REPEAT
15      faktorial:=faktorial*indeks_hitung;
16      indeks_hitung:=indeks_hitung+1;
17    UNTIL indeks_hitung=bilangan+1;
18    writeln('faktorial dari ',bilangan,' adalah ',faktorial:0:0);
19    write('berikan suatu bilangan lagi');
20    readln(bilangan);
21  end;
22  writeln('Ok, Saya keluar ya');
23  end.
```

1.2 *Array*

Sebelumnya, telah dipelajari cara menyimpan suatu masukan (***input***) ke dalam peubah/peubah. Namun jika ingin menyimpan suatu data yang lebih dari satu masukan, maka dapat memanfaatkan peubah larik (***array***). Sebagai contoh, ingin menyimpan lebih dari empat nama mahasiswi seperti pada tabel berikut.

Tabel 1.1: Contoh Data Bertipe ***Array***

nama[1]	nama[2]	nama[3]	nama[4]	···
Dewi Apriani	Dewi Satriani	Dewi Susi Anggraeni	Dewi Dewi	···

Pendeklarasiannya dalam bahasa Pascal yaitu diletakkan setelah pernyataan ***VAR***, lalu diikuti oleh nama ***array***, kapasitas ***array*** dan tipe elemen dalam ***array***. Ingat bahwa dalam bahasa Pascal, peubah ***array*** dideklarasikan dalam tipe yang sama. Misalkan ingin memasukkan peubah ***array*** dengan nama "nama" yang bertipe ***string*** seperti pada Tabel 1.1, maka sintaks deklarasi peubahnya adalah sebagai berikut.

```
1 VAR
2    nama: ARRAY[1..10] of STRING;
```

Perhatikan bahwa peubah "nama" dapat memuat hingga 10 kalimat ***string***, dan indeks dari ***array*** dimulai dari 1 sampai 10.

Contoh lain dalam penggunaan peubah ***array*** adalah sebagai berikut.

```
1  proGRAm berat;
2  uses crt;
3  var   nama_panggilan:aRRay[1..100] of string;
4  berat_badan:array[1..100] of real;
5  n:integer;
6  hitung:real;
7
8  begin
9  clrscr;
10 hitung:=0;
11 for n:=1 to 5 do
12 begin
13 write('Siapa namanya ? ');
14 read(nama_panggilan[n]);
15 write('Berapa beratnya? ');
16 readln(berat_badan[n]);
17 hitung:=hitung+berat_badan[n];
18 end;
19 clrscr;
20
21 for n:=1 to 5 do
22 begin
23 writeln('Namanya adalah ',nama_panggilan[n]);
24 writeln('beratnya ',berat_badan[n]:3:1,'kg loh');
25 end;
26 writeln;
27 writeln('Sehingga berat rata-ratanya adalah ',hitung/n:3:2,'kg.');
28 readln;
29 END.
```

Perhatikan bahwa indeks elemen suatu peubah ***array*** yaitu berada dalam kurung siku. Peubah dengan nama "hitung" digunakan untuk menghitung jumlah seluruh nilai dalam peubah ***array*** "berat", yang digunakan untuk menghitung nilai rata-rata.

Jika suatu aplikasi yang membutuhkan struktur lebih rumit, maka biasanya dibutuhkan peubah ***array*** yang dua dimensi. Peubah ***array*** "berat" pada materi sebelum adalah contoh ***array*** 1 dimensi. Dalam Matematika, ***array*** 1 dimensi disebut vektor, sementara ***array*** 2 dimensi disebut matriks. Misalkan ingin menyimpan nilai praktikum, nilai uts, dan nilai uas, untuk setiap mahasiswa yang mengambil mata kuliah Pemrograman Komputer I, yang dapat disajikan ke dalam Tabel 1.2.

Tabel 1.2: Contoh Data Bertipe ***Array*** Dua Dimensi

Nomor_absen	Praktikum	UTS	UAS
1	78.1	79	80
2	77.0	80	78
3	68.9	75.4	81.2
⋮	⋮	⋮	⋮
100	79.7	82.9	81.3

Perhatikan bahwa indeks ***array*** pada Tabel 1.2 dinyatakan dengan nomor absen. Setiap pemegang nomor absen, memiliki nilai praktikum, nilai uts, dan nilai uas. Untuk merepresentasikan data tersebut, menggunakan kurung siku sebanyak 2 kali. Misalkan ingin memasukkan nilai UTS untuk pemegang nomor absen ke 3, maka sintaksnya adalah sebagai berikut.

```
1 nilai_mahasiswa[3,2]:=75.4;
```

atau dapat juga menggunakan penulisan berikut.

```
1 nilai_mahasiswa[3][2]:=75.4;
```

Secara umum, untuk mendeklarasikan ***array*** 2 dimensi adalah sebagai berikut.

```
1 VAR
2 nama_array:ARRAY[indeks_rentang_1,indeks_rentang_2] OF tipe_elemen;
```

Selanjutnya perhatikan program berikut.

```
1 program penilaian;
```

```
2  uses crt;
3  VAR
4     nilai_mahasiswa: ARRAY[1..100,1..3] OF REAL;
5     jumlah_prak, jumlah_uts, jumlah_uas: REAL;
6     rerata_prak, rerata_uts, rerata_uas: REAL;
7     i,n:INTEGER;
8  begin
9     clrscr;
10    jumlah_prak:=0;
11    jumlah_uts:=0;
12    jumlah_uas:=0;
13
14    write('Berapa mahasiswa? '); readln(n);
15    writeln;
16
17    for i:=1 to n do
18    BEGIN
19      write('Nilai praktikum mahasiswa ke ',i,' = ');
20      readln(nilai_mahasiswa[i][1]);
21      write('Nilai UTS mahasiswa ke ',i,'       = ');
22      readln(nilai_mahasiswa[i][2]);
23      write('Nilai UAS mahasiswa ke ',i,'       = ');
24      readln(nilai_mahasiswa[i][3]);
25      writeln;
26    END;
27
28    for i:=1 to n do
29    BEGIN
30      jumlah_prak := jumlah_prak + nilai_mahasiswa[i][1];
31      jumlah_uts := jumlah_uts + nilai_mahasiswa[i][2];
32      jumlah_uas := jumlah_uas + nilai_mahasiswa[i][3];
33    END;
34
35    rerata_prak:=jumlah_prak/n;
36    rerata_uts:=jumlah_uts/n;
37    rerata_uas:=jumlah_uas/n;
38
39    clrscr;
40    write('Nilai rata-rata praktikum kelas ');
41    writeln('adalah ',rerata_prak:3:2);
42    writeln('Nilai rata-rata UTS kelas adalah ',rerata_uts:3:2);
43    writeln('Nilai rata-rata UAS kelas adalah ',rerata_uas:3:2);
```

```
44
45      readln;
46  end.
```

Contoh lain penggunaan ***array*** 2 dimensi adalah sebagai berikut.

```
1   program penilaian2;
2   uses crt;
3   VAR
4      nilai_mahasiswa: ARRAY[1..100,1..3] OF REAL;
5      rata_rata_mahasiswa: ARRAY[1..100] OF REAL;
6      jumlah_nilai:REAL;
7      i,j,n:INTEGER;
8   begin
9      clrscr;
10
11     write('Berapa mahasiswa? '); readln(n);
12     writeln;
13
14     for i:=1 to n do
15     BEGIN
16       write('Nilai praktikum mahasiswa ke ',i,' = ');
17       readln(nilai_mahasiswa[i][1]);
18       write('Nilai UTS mahasiswa ke ',i,'       = ');
19       readln(nilai_mahasiswa[i][2]);
20       write('Nilai UAS mahasiswa ke ',i,'       = ');
21       readln(nilai_mahasiswa[i][3]);
22       writeln;
23     END;
24
25     for i:=1 to n do
26     BEGIN
27       jumlah_nilai:=0;
28       for j:=1 to 3 do
29       BEGIN
30         jumlah_nilai:=jumlah_nilai+nilai_mahasiswa[i,j];
31         rata_rata_mahasiswa[i]:=jumlah_nilai/3;
32       END;
33     END;
34
35     writeln;
```

```
36      writeln('  ','no absen','  ','rata-rata');
37      writeln('-------------------------------------------');
38      for i:=1 to n do
39      BEGIN
40      writeln('  ', i:4      ,'  ',rata_rata_mahasiswa[i]:12:2);
41      END;
42
43      readln;
44  end.
```

Pendeklarasian ***array*** dapat dilakukan dengan menulisnya dalam bagian ***Type***. Perhatikan sintaks berikut.

```
1  TYPE
2     tipe_array = ARRAY[1..8] of INTEGER;
3  VAR
4     array_ku, array_mu: tipe_array;
```

Contoh selanjutnya adalah ***array*** 2 dimensi. Kemungkinan lain dalam mendeklarasikannya adalah sebagai berikut.

```
1  TYPE
2     rentang_ku = 1..8;
3     rentang_mu = 1..100;
4     tipe_array = ARRAY[rentang_ku,rentang_mu] of INTEGER;
5  VAR
6  array_ku, array_mu, array_nya: tipe_array;
```

1.3 Prosedur dan Fungsi

Dalam pembuatan aplikasi nyata, sering ditemui masalah yang sangat kompleks. Sehingga diperlukan pembagian dari program utama menjadi perintah yang lebih sederhana, lalu dimasukkan ke dalam sub program. Hal ini dilakukan untuk menghindari adanya kesalahan dalam pembuatan aplikasi nyata. Pembuatan sub-sub program inilah yang nantinya akan dikombinasikan menjadi suatu program yang

utuh. Dalam bahasa Pascal, penguraian menjadi sub-sub program yang lebih kecil ini dapat berupa prosedur atau fungsi.

1.3.1 Prosedur

Perhatikan program berikut yang dapat digunakan untuk menentukan nilai terkecil jika diberikan tiga bilangan.

```
1  program sebelum_prosedur;
2  uses crt;
3  var
4     a, b, c,  minim: integer;
5
6  BEGIN
7     CLRSCR;
8     writeln('Tulis tiga bilangan: ');
9     readln(a, b, c);
10    if a < b then
11       minim:= a
12    else
13       minim:= b;
14
15    if c < minim then
16       minim:= c;
17
18    writeln('Minimum: ', minim);
19    readln;
20 END.
```

Algoritma dalam mencari nilai terkecil tersebut, dapat dijadikan sub program. Perhatikan program berikut.

```
1  program sesudah_prosedur;
2  uses crt;
3  var
4     a, b, c,  minim: integer;
5
6  procedure carimin(x, y, z: integer; var m: integer);
```

```
7  begin
8     if x < y then
9        m:= x
10    else
11       m:= y;
12
13    if z < m then
14       m:= z;
15 end;
16
17 BEGIN
18    CLRSCR;
19    writeln('Tulis tiga bilangan: ');
20    readln(a, b, c);
21    carimin(a, b, c, minim);
22
23    writeln('Minimum: ', minim);
24    readln;
25 END.
```

Prosedur "carimin" memerlukan tiga nilai masukan bilangan dari *keyboard*. Variabel "m" dalam prosedur "carimin" berperan sebagai hasil dari pencarian nilai terkecil. Sehingga pemanggilan prosedur "carimin" menggunakan empat argumen. Namun dalam praktiknya, penggunaan suatu prosedur tidak harus memerlukan argumen. Contoh berikut adalah program yang memiliki sub program yang tanpa menggunakan argumen.

```
1  Program pilih_mata_kuliah;
2  Uses Crt;
3  Var
4     kode_mk: byte;
5
6  Procedure menunya;
7  Begin
8     CLRSCR;
9     writeln('Selamat Datang');
10    writeln('Silahkan pilih mata kuliah berikut:');
11    writeln('1 - Kalkulus Elementer');
12    writeln('2 - Geometri Analitik');
13    writeln('3 - Fungsi Kompleks I');
```

```
14     writeln('4 - Struktur Aljabar I');
15     writeln('5 - Teori Graf');
16     writeln('6 - Proses Stokastik');
17  End;
18
19  Procedure ygdipilih(namanya: string; sks: integer);
20  Begin
21     Writeln('Anda memilih mata kuliah ', namanya,' (',sks,' SKS)');
22  End;
23
24  BEGIN
25     menunya;
26     write('Pilihan Anda = ');
27     readln(kode_mk);
28
29     CASE kode_mk OF
30     1: ygdipilih('Kalkulus Elementer', 3);
31     2: ygdipilih('Geometri Analitik', 3);
32     3: ygdipilih('Fungsi Kompleks I', 3);
33     4: ygdipilih('Struktur Aljabar I', 2);
34     5: ygdipilih('Teori Graf', 2);
35     6: ygdipilih('Proses Stokastik', 3);
36     ELSE
37        Writeln('Anda salah memilih');
38     END;
39     write('Tekan ENTER untuk menutup program');
40     readln;
41  End.
```

Dalam hal ini, prosedur "menunya" tidak memerlukan argumen. Perlu diingat bahwa jika suatu prosedur menggunakan argumen, maka setiap peubahnya selalu dinyatakan dengan tipe data yang sesuai. Perhatikan bahwa prosedur "menu" ditulis untuk menampilkan pilihan mata kuliah yang akan diambil, atau dengan kata lain bahwa prosedur ini memuat perintah ***CLRSCR*** dan beberapa perintah ***writeln***. Sehingga prosedur merupakan kumpulan pernyataan atau perintah yang dapat dipanggil.

Prosedur sebagai sub program juga dapat menggunakan argumen yang bertipe ***array***. Perhatikan contoh program berikut.

```
1  Program urutkan;
2  Uses crt;
3  TYPE
4     rentang = 1..3;
5     tipearray = ARRAY[rentang] OF integer;
6  VAR
7     tanyakan_ke_dewi :tipearray;
8
9  { ————————— Prosedur mendata berat badan Dewi ————————— }
10 Procedure mendata(L: integer; var R :tipearray);
11 VAR
12    I :integer;
13 begin
14    FOR I := 1 TO L DO
15    begin
16      write('Berat badan Dewi ke ', I,': ');
17      READLN(R[I])
18    end;
19 end;
20
21 { ———————— Prosedur mengurutkan berat badan Dewi ———————— }
22 Procedure urut(M: integer; var S :tipearray);
23 VAR
24    I, J, sementara :integer;
25 begin
26    FOR I := 1 TO M—1 DO
27    FOR J := I+1 TO M DO
28    IF S[I] > S[J] THEN
29    begin { ditukarkan }
30    sementara := S[J];
31    S[J] := S[I];
32    S[I] := sementara;
33    end;
34 end;
35
36 { —— Prosedur menuliskan berat badan Dewi setelah diurutkan —— }
37 Procedure tulis(N: integer; T :tipearray);
38 VAR
39    I :integer;
40 begin
41    writeln;
```

```
42     write('Berat badan setelah diurutkan : ');
43     FOR I := 1 TO N DO
44     write(T[I],' ');
45     writeln;
46 end;
47
48 { ———————————————— Program Utama ———————————————— }
49 BEGIN
50     ClrScr;
51     mendata(3, tanyakan_ke_dewi);
52     urut(3, tanyakan_ke_dewi);
53     tulis(3, tanyakan_ke_dewi);
54     writeln('Tekan ENTER untuk menutup program');
55     readln;
56 END.
```

1.3.2 Fungsi

Fungsi merupakan sub program yang menghasilkan suatu nilai. Sama halnya dengan prosedur, fungsi juga dapat memiliki argumen. Walaupun prosedur tidak mengharuskan adanya argumen. Contoh sederhana penggunaannya adalah fungsi yang dapat menambahkan, mengalikan, mengurangkan, dan membagikan jika diberikan dua buah bilangan. Perhatikan contoh program berikut.

```
1  Program fungsisederhana;
2  uses crt;
3
4  function tambah(a,b : integer):integer;
5  begin
6    tambah:=a+b;
7  end;
8
9  function kali(a,b : integer):longint;
10 begin
11   kali:=a*b;
12 end;
13
14 function kurang(a,b : integer):integer;
```

```
15 begin
16   kurang:=a-b;
17 end;
18
19 function bagi(a,b : integer):real;
20 begin
21   bagi:=a/b;
22 end;
23
24 var
25   a,b: integer;
26
27 begin
28    clrscr;
29    Write('a = '); readln(a);
30    Write('b = '); readln(b);
31    Writeln;
32    Writeln('a+b = ',tambah(a,b));
33    Writeln('a-b = ',kurang(a,b));
34    Writeln('a*b = ',kali(a,b));
35    Writeln('a/b = ',bagi(a,b):4:2);
36    Writeln;
37    Writeln('Terimakasih');
38    readln;
39 end.
```

Dalam program bernama "fungsisederhana" ini, argumen yang digunakan adalah peubah "a" dan "b" yang masing-masing bertipe ***integer***. Nilai yang ingin dihasilkan dari beberapa fungsi dalam program tersebut bervariasi, yakni tergantung pada fungsi yang ingin digunakan.

Fungsi dapat dimanfaatkan untuk membuat suatu tabel sinus. Perhatikan contoh program berikut.

```
1 PROGRAM fungsisinus;
2 uses crt;
3 var
4   i,n:integer;
5
6 function fungsisin(x: integer): REAL;
```

```
7  begin
8    fungsisin:=sin(x*pi/180);
9  end;
10
11 BEGIN
12   CLRSCR;
13   Writeln('============');
14   Writeln(' x    sin(x)');
15   Writeln('--------------');
16   for i:=1 to 18 do
17   begin
18     writeln(i*10:3,'    ',fungsisin(i*10):5:2);
19   end;
20   Writeln('============');
21   readln;
22 END.
```

Contoh lain dari penggunaan fungsi adalah menampilkan barisan Fibonacci yang dikenalkan oleh Leonardo Fibonacci. Barisan ini adalah sekumpulan bilangan yang merupakan penjumlahan dari dua bilangan sebelumnya. Berikut adalah contoh programnya.

```
1  PROGRAM bilanganFibonacci;
2  USES crt;
3  CONST
4      BERI_SPASI = '       ';
5  VAR
6      i,n : integer;
7
8  FUNCTION Fibonacci(i: INTEGER): LONGINT;
9  begin
10     if i ≤ 1 then
11     Fibonacci := 1
12     else
13     Fibonacci := Fibonacci(i-1) + Fibonacci(i-2);
14 end;
15
16 BEGIN
17     CLRSCR;
18
```

```
19      write('Banyaknya bilangan Fibonacci: ');
20      readln(n);
21      writeln('Suku ke', BERI_SPASI, 'Bilangan Fibonacci');
22
23      i := 1;
24      WHILE i < n+1 DO
25      BEGIN
26         writeln('  ',i:2, BERI_SPASI, Fibonacci(i):8);
27         i := i + 1;
28      END;
29      writeln;
30      writeln('Tekan ENTER untuk menutup program ini...');
31      readln;
32  END.
```

Fungsi "Fibonacci" tersebut menggunakan satu argumen yang bertipe ***integer***. Untuk mengantisipasi bilangan Fibonacci yang besar, maka program ini menggunakan hasil yang bertipe ***LONGINT***.

Selain contoh-contoh yang telah diberikan, penggunaan fungsi juga dapat memiliki sifat komposisi, yakni penggunaan suatu fungsi dalam fungsi lain (bersarang). Perhatikan contoh program berikut.

```
1  Program menghitung_volume_balok;
2  uses crt;
3  var p,l,t,v:real;
4
5  function hitungluasalasnya(p,l: real):real;
6  begin
7     hitungluasalasnya:=p*l;
8  end;
9
10 function hitungvolume(luas_alas,t: real):real;
11 begin
12    hitungvolume:=luas_alas*t;
13 end;
14
15 BEGIN
16    ClrScr;
17    Writeln('Program ini dapat menghitung volume balok.');
```

```
    Writeln('Ingat bahwa rumus volume balok adalah V=p*l*t.');
    Writeln;

    Writeln('Berapa panjang baloknya ?');
    Write('p = '); readln(p);
    Writeln('Berapa lebar baloknya ?');
    Write('l = '); readln(l);
    Writeln('Berapa tinggi baloknya ?');
    Write('t = '); readln(t);
    Writeln;
    Writeln('¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬');

    v:=hitungvolume(hitungluasalasnya(p,l),t);
    Writeln('V = ',p:2:2,'*',l:2:2,'*',t:2:2,' = ',v:4:3);
    Writeln('Maka, volume balok adalah ',v:4:3);
    Readln;
END.
```

Program bernama "menghitung_volume_balok" dapat menghitung volume balok jika diketahui panjang, lebar, dan tinggi suatu balok. Program ini menggunakan dua fungsi, yakni "hitungluasalasnya" dan "hitungvolume". Fungsi "hitungluasalasnya" memiliki argumen "p" dan "l" yang masing-masing berperan sebagai panjang dan lebar suatu balok. Luas alas balok tersebut dapat dihitung dengan memanfaatkan fungsi "hitungluasalasnya" ini. Setelah hasil perhitungan luas alas diperoleh, maka selanjutnya menghitung volume dengan memanfaatkan fungsi "hitungvolume". Argumen yang diperlukan dalam menggunakan fungsi "hitungvolume" adalah luas alas dan tinggi dari balok. Sehingga dengan menggunakan sifat dekomposisi dari fungsi, maka argumen luas alas dapat diperoleh dari fungsi "hitungluasalasnya" yang telah dihitung sebelumnya.

1.4 *Record*

Suatu *record* adalah koleksi *item* data yang saling berhubungan. Setiap *item* dalam *record* disebut *field*. Tidak seperti *array*, suatu *record* dapat berisi *field* yang tipe data yang berbeda. Perhatikan Tabel 1.3, yang merupakan beberapa identitas seorang mahasiswa.

Tabel 1.3: *Record* Mahasiswa

Field ke	Informasi	Tipe Data
1	Nama	String
2	Usia	Byte
3	Jenis_kelamin	Char
4	Jurusan	String
5	NIM	String
6	IPK	Real

Tidak seperti *array* (yang setiap indeksnya tidak boleh ditukar), posisi informasi dalam suatu *record* dapat diubah. Perhatikan Tabel 1.4 berikut.

Tabel 1.4: *Record* Mahasiswa Versi Lain

Field ke	Informasi	Tipe Data
1	Jenis_kelamin	Char
2	Usia	Byte
3	NIM	String
4	Nama	String
5	Jurusan	String
6	IPK	Real

Dalam *record*, informasi pada Tabel 1.3 dianggap sama saja dengan informasi pada Tabel 1.4. Hal ini juga berdampak pada proses pengisian data (elemen) *field* untuk *record*, dimana pengisiannya diperbolehkan tidak sesuai urutan pendeklarasian *record*.

Selanjutnya adalah cara pendeklarasian tipe *record*. Adapun pendeklarasian tipe *record* untuk identitas Mahasiswa seperti pada Tabel 1.3 adalah sebagai berikut.

```
1  TYPE
2  RecordMahasiswa = RECORD
3                      Nama          : String;
4                      Usia          : Byte;
5                      Jenis_kelamin : Char;
6                      Jurusan       : String[20];
7                      NIM           : String[10];
8                      IPK           : Real;
9                    END;
10 VAR   IdentitasMahasiswa : RecordMahasiswa;
```

Untuk lebih jelasnya, perhatikan contoh program berikut.

```
1  Program DataMahasiswa;
2  Uses crt;
3  TYPE
4  RecordMahasiswa = RECORD
5                      Nama          : String;
6                      Usia          : Byte;
7                      Jenis_kelamin : Char;
8                      Jurusan       : String[20];
9                      NIM           : String[10];
10                     IPK           : Real;
11                   END;
12
13 var
14   IdentitasMahasiswa : RecordMahasiswa;
15   jk: String;
16
17 BEGIN
18   ClrScr;
19   IdentitasMahasiswa.Nama:='Sandra Dewi';
20   IdentitasMahasiswa.Usia:=19;
21   IdentitasMahasiswa.Jenis_Kelamin:='P';
22   IdentitasMahasiswa.Jurusan:='Matematika';
23   IdentitasMahasiswa.NIM:='1707065000';
24   IdentitasMahasiswa.IPK:=3.99;
25
26   if IdentitasMahasiswa.Jenis_Kelamin='L' then jk:='Laki-laki'
27   else jk:='Perempuan';
28
29   Writeln('Data Mahasiswa Berprestasi');
30   Writeln('Nama Lengkap  = ',IdentitasMahasiswa.Nama);
31   Writeln('Umur          = ',IdentitasMahasiswa.Usia,' tahun');
32   Writeln('Jenis Kelamin = ',jk);
33   Writeln('Jurusan       = ',IdentitasMahasiswa.Jurusan);
34   Writeln('NIM           = ',IdentitasMahasiswa.NIM);
35   Writeln('IPK           = ',IdentitasMahasiswa.IPK:3:2);
36
37   Readln;
38 END.
```

Penggunaan tipe *record* juga dapat memanfaatkan pernyataan ***WITH***. Perhatikan contoh program berikut.

```
1  Program DataMahasiswa2;
2  Uses crt;
3  TYPE
4  RecordMahasiswa = RECORD
5                      Nama          : String;
6                      Usia          : Byte;
7                      Jenis_kelamin : Char;
8                      Jurusan       : String[20];
9                      NIM           : String[10];
10                     IPK           : Real;
11                   END;
12
13 var
14   IdentitasMahasiswa : RecordMahasiswa;
15   jk: String;
16
17 BEGIN
18   ClrScr;
19   WITH IdentitasMahasiswa DO
20   BEGIN
21   Nama:='Sandra Dewi';
22   Usia:=19;
23   Jenis_Kelamin:='P';
24   Jurusan:='Matematika';
25   NIM:='1707015000';
26   IPK:=3.99;
27   END;
28
29   if IdentitasMahasiswa.Jenis_Kelamin='L' then jk:='Laki-laki'
30   else jk:='Perempuan';
31
32   WITH IdentitasMahasiswa DO
33   BEGIN
34   Writeln('Data Mahasiswa Berprestasi');
35   Writeln('Nama Lengkap  = ',Nama);
36   Writeln('Umur          = ',Usia,' tahun');
37   Writeln('Jenis Kelamin = ',jk);
38   Writeln('Jurusan       = ',Jurusan);
```

```
39    Writeln('NIM           = ',NIM);
40    Writeln('IPK           = ',IPK:3:2);
41    END;
42
43    Readln;
44  END.
```

Tipe *record* dalam suatu program, juga dapat bersarang (*nested*). Perhatikan contoh program berikut.

```
1   Program DataMahasiswa3;
2   Uses crt;
3   TYPE
4   RecordNama      = RECORD
5                       Nama_depan    : String[15];
6                       Nama_tengah   : String[15];
7                       Nama_belakang : String[15];
8                       Nama_panggilan: String[10];
9                     END;
10
11  RecordMahasiswa = RECORD
12                      Nama          : RecordNama;
13                      Usia          : Byte;
14                      Jenis_kelamin : Char;
15                      Jurusan       : String[20];
16                      NIM           : String[10];
17                      IPK           : Real;
18                    END;
19
20  var
21    IdentitasMahasiswa : RecordMahasiswa;
22    jk: String;
23
24  BEGIN
25    ClrScr;
26    WITH IdentitasMahasiswa DO
27    BEGIN
28      WITH Nama DO
29      BEGIN
30        Nama_depan:='Dewi';
```

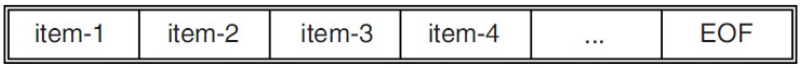
```
31        Nama_tengah:='Susi';
32        Nama_belakang:='Anggraeni';
33        Nama_panggilan:='Desus';
34      END;
35    Usia:=19;
36    Jenis_Kelamin:='P';
37    Jurusan:='Matematika';
38    NIM:='1707015000';
39    IPK:=3.99;
40    END;
41
42    if IdentitasMahasiswa.Jenis_Kelamin='L' then jk:='Laki-laki'
43    else jk:='Perempuan';
44
45    Writeln('Data Mahasiswa Berprestasi');
46    WITH IdentitasMahasiswa DO
47    BEGIN
48      WITH Nama DO
49      BEGIN
50      Writeln('Nama Lengkap  = ',Nama_depan,' ',
51        Nama_tengah,' ',Nama_belakang);
52      Writeln('Nama Panggilan= ',Nama_panggilan);
53      END;
54    Writeln('Umur          = ',Usia,' tahun');
55    Writeln('Jenis Kelamin = ',jk);
56    Writeln('Jurusan       = ',Jurusan);
57    Writeln('NIM           = ',NIM);
58    Writeln('IPK           = ',IPK:3:2);
59    END;
60
61    Readln;
62  END.
```

1.5 *Data Files*

Suatu data tidak hanya dapat diperoleh dalam proses pemasukan data melalui *keyboard* saja (menggunakan pernyataan ***read***), tetapi juga dapat melalui *file* yang terpisah dari program Pascal. Hal ini bermanfaat untuk menganalisis data yang

telah diperoleh sebelumnya.

Secara umum, *file* merupakan kumpulan *item* yang tersimpan dalam *harddisk* atau penyimpanan eksternal lainnya, yang tersusun dalam rangkaian yang ditunjukkan dalam Gambar 1.1.



Gambar 1.1: Ilustrasi kumpulan *item* dalam suatu *file*

Pascal mengenali dua tipe *file*, yakni *file* ***TEXT*** dan *non-**TEXT** file*. Materi ini hanya membahas tipe pertama. ***TEXT*** *file* merupakan struktur *file* yang lebih sederhana, karena semua elemennya bertipe karakter. Perhatikan struktur standar ***TEXT*** *file* berikut.

Ini adalah contoh isi dalam ***TEXT*** *file*. (**EOLN**)
Setiap baris merupakan kumpulan data bertipe ***CHAR***. (**EOLN**)
Setiap baris dipisah oleh tanda *end-of-line*. (**EOLN**)
Setiap *File* diakhiri oleh tanda *end-of-file*. (**EOF**)

Peubah ***TEXT*** *file* dideklarasikan sebagai data bertipe ***TEXT***. Prosedur standar untuk menghubungkan *file* tersebut dengan program Pascal adalah ***ASSIGN***. Untuk membacanya, harus dibuka menggunakan prosedur standar ***RESET***. Lalu diakhiri dengan menutupnya menggunakan prosedur standar ***CLOSE***. Perhatikan sintaks standar berikut.

```
1 VAR
2    fileTEXT : TEXT;
3 begin
4    ASSIGN(fileTEXT, 'lokasi file'); {menghubungkan file}
5    RESET(fileTEXT); {membuka file}
6    CLOSE(fileTEXT); {menutup file}
7 end.
```

Berikut adalah contoh program yang dapat mengakses suatu *file* dalam komputer.

```
1 Program bacatxt;
2 uses crt;
```

```
3  const
4     nama_filenya = 'D:\kalimat.txt';
5  var
6     baca_sbg_teks : TEXT;
7     karakter      : CHAR;
8
9  begin
10 CLRSCR;
11 assign(baca_sbg_teks, nama_filenya);
12 reset(baca_sbg_teks);
13 WHILE NOT EOF(baca_sbg_teks) DO
14 begin
15   WHILE NOT EOLN(baca_sbg_teks) do
16   begin
17     read(baca_sbg_teks,karakter); {membaca setiap karakter}
18     write(karakter); {menulis karakternya}
19   end;
20   readln(baca_sbg_teks);
21   writeln;
22 end;
23
24 CLOSE(baca_sbg_teks);
25 writeln;
26 writeln('Tekan ENTER untuk menutup program ini');
27 readln;
28 end.
```

Fungsi ***EOF*** dalam program bernama "bacatxt" menyatakan suatu karakter yang posisinya berada pada akhir dari isi dalam *file*, sedangkan fungsi ***EOLN*** menyatakan karakter yang posisinya berada pada akhir dari suatu baris. Jika *file* yang diberikan terdiri dari beberapa baris, maka perlu ditambahkan pernyataan ***readln*** dan diikuti oleh pernyataan ***writeln***.

Program Pascal juga dapat membaca ***TEXT*** *file* menggunakan tipe ***STRING***. Setiap ***STRING*** tersebut memiliki panjang maksimal 80 karakter, yang merupakan banyaknya karakter dari suatu baris yang diharapkan. Setiap akhir baris, posisi *file pointer* berpindah ke baris selanjutnya. Perhatikan contoh program berikut.

```
1  Program bacatxt_baris;
```

```
2  Uses Crt;
3  Var
4    File_ku : TEXT;
5    Baca_satu_baris, Nama_file : STRING[80];
6
7  BEGIN
8    Clrscr;
9    Writeln('Masukan alamat dan nama file yang akan ditampilkan:');
10   Readln(Nama_file);
11   Writeln;
12   Writeln('Isi dari file ',Nama_file,' adalah sebagai berikut: ');
13   Writeln('-------------------------------------------------------------');
14
15   ASSIGN(File_ku, Nama_file);
16   RESET(File_ku);
17   WHILE NOT EOF(File_ku) DO
18     Begin
19       Readln(File_ku,Baca_satu_baris);
20       Writeln(Baca_satu_baris);
21     End;
22   CLOSE(File_ku);
23   Writeln('-------------------------------------------------------------');
24   Writeln;
25   Write('Tekan ENTER untuk menutup program ini');
26   Readln;
27 END.
```

Program yang bernama "bacatxt_baris" tersebut dapat membaca ***TEXT*** *file* yang ditentukan oleh pengguna program, dengan cara menuliskan alamat, nama, dan ekstensi dari *file* yang ingin dibaca. Berbeda dengan contoh program sebelumnya (yang bernama "bacatxt"), program ini menggunakan perintah ***readln*** dan ***writeln*** untuk membaca dan menampilkan isi dari *file*.

Jika ingin membuat suatu *file*, maka gunakan prosedur standar ***REWRITE***. Lalu beberapa baris bertipe ***STRING*** yang ingin dimasukan ke dalam *file* tersebut, dimasukan melalui prosedur standar ***WRITE*** atau ***WRITELN***. Perhatikan contoh program sederhana berikut.

```
1  Program membuat_file;
```

```
2  Const
3    kalimatnya = 'Saya suka Matematika';
4  var
5    F :TEXT;
6
7  Begin
8    ASSIGN(F, 'D:\suka_mtk.txt');
9    REWRITE(F);
10   Writeln(F, kalimatnya);
11   CLOSE(F);
12 End.
```

Program dengan nama "membuat_file" memberikan hasil berupa ***TEXT*** *file* yang ditaruh dengan alamat "D:\suka_mtk.txt". Contoh program selanjutnya adalah membuat *record file* dari identitas Mahasiswa. Perhatikan program berikut.

```
1  Program membuat_file_record;
2  Uses Crt;
3  Type
4    RecordNama = RECORD
5                   Nama_depan : String[15];
6                   Nama_tengah : String[15];
7                   Nama_belakang : String[15];
8                   Nama_panggilan: String[10];
9                 END;
10
11   RecordMahasiswa = RECORD
12                   Nama : RecordNama;
13                   Usia : Byte;
14                   Jenis_kelamin : Char;
15                   Jurusan : String[20];
16                   NIM : String[10];
17                   IPK : Real;
18                 END;
19
20 var
21 IdentitasMahasiswa : RecordMahasiswa;
22 F :TEXT;
23
24 Begin
```

```
25    ClrScr;
26    ASSIGN(F, 'D:\record_mhs.txt');
27    REWRITE(F);
28    WITH IdentitasMahasiswa DO
29      WITH Nama DO
30      Begin
31        Writeln('Program ini akan memasukkan identitas Anda,');
32        Writeln('ke dalam file D:\record_mhs.txt');
33        Readln;
34        Writeln;
35        Write('NIM Anda                  = '); Readln(NIM);
36        Write('Nama:       depan = '); Readln(Nama_depan);
37        Write('           tengah = '); Readln(Nama_tengah);
38        Write('         belakang = '); Readln(Nama_belakang);
39        Write('        panggilan = '); Readln(Nama_panggilan);
40        Write('Jurusan Anda              = '); Readln(Jurusan);
41        Write('Jenis Kelamin Anda (L/P) = '); Readln(Jenis_kelamin);
42        Write('Usia Anda                 = '); Readln(Usia);
43        Write('IPK Anda                  = '); Readln(IPK);
44
45        Writeln(F, NIM);
46        Write(F, Nama_depan);
47        Write(F, ' ', Nama_tengah);
48        Writeln(F, ' ', Nama_belakang);
49        Writeln(F, Nama_panggilan);
50        Writeln(F, Jurusan);
51        Writeln(F, Jenis_kelamin);
52        Writeln(F, Usia);
53        Writeln(F, IPK:3:2)
54      End;
55    CLOSE(F);
56  End.
```

Daftar Pustaka

Abolrous, S.A. (2000). *Learn Pascal*. Wordware Publishing, Inc.