**MODUL LATIHAN**
**PEMROGRAMAN DASAR**

# BAHASA PASCAL
# DENGAN FREEPASCAL

**v.1.0**
**DOKUMEN UNTUK UMUM**
**TIDAK UNTUK DIPERJUALBELIKAN**

**oleh:**
**BISMA JAYADI**

**website:** http://beeography.modblog.com
**email:** bisma_j@yahoo.com

# LATIHAN I
# PERULANGAN DAN SELEKSI KONDISI

## 1.1. Pendahuluan

Bahasa Pascal adalah bahasa pemrograman tingkat tinggi (*high level language*) yang cukup populer, khususnya di Indonesia. Hal ini disebabkan bahasa Pascal lebih mudah dipahami dibanding bahasa pemrograman lainnya, seperti bahasa C, bahasa *assembler*, dan lain sebagainya. Selain itu, bahasa Pascal adalah bahasa pemrograman yang terstruktur dan lebih mendekati bahasa manusia (bahasa Inggris) sehingga sangat cocok diterapkan dalam dunia pendidikan. Dalam latihan ini, digunakan *compiler* FreePascal (www.freepascal.org) yang bersifat *open source* dan tersedia di banyak *operating system* (DOS, Windows, Linux, Macintosh, FreeBSD, dan lain sebagainya).

Sebelum mempelajari pemrograman dengan bahasa Pascal, sebaiknya mengerti terlebih dahulu tentang konsep dan teknik pemrograman. Konsep pemrograman adalah bentuk dasar dari suatu program yaitu algoritma program. Algoritma adalah urutan proses yang dilakukan oleh sebuah program, umumnya algoritma ini berbentuk *flowchart* (diagram alir). Teknik pemrograman adalah cara mengubah suatu algoritma menjadi program yang sebenarnya dengan bahasa pemrograman tertentu. Konsep dan teknik pemrograman adalah dasar dari pemrograman komputer, dengan menguasai kedua hal tersebut maka mempelajari pemrograman menjadi lebih mudah.

## 1.2. Struktur Program

Bahasa Pascal, secara garis besar mempunyai struktur sebagai berikut :

1. Judul program,
2. Blok program yang terdiri dari :
   a. Bagian deklarasi, meliputi :
      - deklarasi label,
      - deklarasi konstanta,
      - deklarasi tipe,
      - deklarasi variabel,
      - deklarasi prosedur dan/atau fungsi.
   b. Bagian pernyataan, berisi perintah-perintah.

Untuk lebih jelas memahami struktur program Pascal, perhatikan contoh program berikut ini :

```
program Struktur_Pascal;     { judul program }

label                        { deklarasi label }
  ulang;

const                        { deklarasi konstanta }
  bahasa = 'Pascal';
  versi  = 1.9;

type                         { deklarasi tipe }
  tipeku = integer;

var                          { deklarasi variabel }
  A : tipeku;

procedure Tulis;             { deklarasi prosedur }
begin
  Writeln('Latihan Pascal 1: Perulangan dan Seleksi Kondisi');
  Writeln('-----------------------------------------------');
  Writeln;
  Writeln('Halo,saya sedang belajar bahasa ',bahasa,' versi ',versi);
end;

begin                        { bagian pernyataan }
  Tulis;
  Readln;
end.
```

## 1.3. Perulangan

Perulangan (*iterasi*) adalah proses yang berulang. Iterasi selalu ada dalam bahasa pemrograman apapun, karena disinilah letak kelebihan komputer dibanding manusia, yaitu mampu melakukan hal yang sama berulang kali tanpa kesalahan akibat bosan atau lelah. Dengan perulangan, program menjadi lebih pendek dan sederhana.

Dalam Pascal dikenal tiga macam perintah (*statement*) perulangan, yaitu *statement* `for…do`, `repeat…until` dan `while…do`. Perulangan `for…do` adalah perulangan dengan penghitung (*counter*), perulangan `repeat…until` adalah perulangan dengan syarat akhir sedang perulangan `while…do` adalah perulangan dengan syarat awal.

Untuk lebih jelas memahami proses perulangan ini, perhatikan contoh program di bawah ini :

```
program Iterasi1;

var
  I, Data     : integer;
  Nilai, Rata : real;
  Jumlah      : real;
```

```
begin
  Writeln('Latihan Pascal 1: Perulangan dan Seleksi Kondisi');
  Writeln('-----------------------------------------------');
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;

  Jumlah := 0;
  Writeln;
  Write('Masukkan jumlah data : ');
  Readln(Data);
  Writeln;

  for I := 1 to Data do
  begin
    Write('Masukkan data ke-',I,' : ');
    Readln(Nilai);
    Jumlah := Jumlah + Nilai;
  end;

  Rata := Jumlah/Data;

  Writeln;
  Writeln('Rata-ratanya = ',Rata:6:3);
  Readln;
end.
```

Untuk lebih memahami apa yang dilakukan oleh program, jalankan program dengan tombol F7 dan amati perubahan variabelnya (cara mengamati perubahan variabel dapat ditanyakan pada asisten). Jika telah mengerti, buatlah program yang serupa dengan menggunakan *statement* perulangan yang lain.

## 1.4. Seleksi Kondisi

Seleksi kondisi adalah proses penentuan langkah berikutnya berdasarkan proses yang terjadi sebelumnya. Seleksi kondisi ini sangat penting dalam pemrograman sebab dengan adanya seleksi kondisi, program dapat menentukan proses apa yang harus dilakukan selanjutnya berdasarkan keadaan sebelumnya. Sehingga nampak seolah-olah program dapat berpikir dan mengambil keputusan. Disinilah letak kekurangan komputer yaitu tidak mampu berpikir sendiri, semua hal yang dilakukan adalah berdasarkan perintah.

Dalam Pascal ada tiga macam perintah seleksi kondisi, yaitu *statement* `if…then`, `if…then…else` dan `case…of`. Seleksi kondisi dengan `if…then` digunakan untuk mengambil satu keputusan diantara dua pilihan sedang seleksi kondisi dengan `if…then…else` dan `case…of` digunakan untuk mengambil satu keputusan diantara banyak pilihan.

Untuk lebih memahami tentang seleksi kondisi, perhatikan contoh program di bawah ini :

```
program Iterasi2;

var
  Nilai : string;

begin
  Writeln('Latihan Pascal 1: Perulangan dan Seleksi Kondisi');
  Writeln('-----------------------------------------------');
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
  Write('Masukkan nilai mata kuliah DKP Anda (huruf) : ');
  Readln(Nilai);

  if Length(Nilai) > 2 then
    Writeln('Nilai tidak mungkin lebih dari dua karakter !');
  else
  begin
    if Nilai = 'A' then
       Writeln('Anda berbakat menjadi programer !')
    else if Nilai = 'B+' then
       Writeln('Anda bisa menjadi programer handal.')
    else if Nilai = 'B' then
       Writeln('Anda mampu menjadi programer.')
    else if Nilai = 'C+' then
       Writeln('Anda cukup mampu menjadi programer.')
    else if Nilai = 'C' then
       Writeln('Anda kurang tertarik pada pemrograman.')
    else if Nilai = 'D+' then
       Writeln('Anda tidak suka pemrograman, ya !')
    else if Nilai = 'D' then
       Writeln('Anda pasti benci lihat program !')
    else if Nilai = 'E' then
       Writeln('Anda pasti nggak niat kuliah !')
    else
       Writeln('Anda salah memasukkan nilai !');
  end;
  Readln;
end.
```

Sebagaimana pada program Iterasi1, jalankan program dengan F7. Kemudian ubah baris perintah seleksi kondisi `if…then…else` setelah `if…then…else` yang pertama menjadi `case…of`, tentunya diikuti pula dengan beberapa perubahan yang lain sehingga program tetap dapat dijalankan.

## 1.5. Latihan

1. Secara umum dalam bahasa pemrograman terdapat tiga macam jenis perulangan. Sebutkan dan jelaskan masing-masing jenis tersebut !

2. Jelaskan apa yang dimaksud dengan pemrograman terstruktur dan jelaskan keuntungan memprogram dengan cara terstruktur !
3. Jelaskan apa yang dimaksud dengan *statement* atau perintah dalam Pascal dan apa pula bedanya dengan *reserved words* !
4. Perhatikan program Iterasi2 yang menggunakan perintah seleksi kondisi `if…then…else`, bandingkan dengan yang menggunakan `case…of` (yang Anda buat saat latihan sebelumnya). Bandingkan dan jelaskan perbedaannya !
5. Perhatikan struktur program Pascal ! Diantara bagian-bagian struktur tersebut, manakah yang harus ada ? Jelaskan mengapa demikian ?

# LATIHAN II
# PROSEDUR DAN FUNGSI

## 2.1. Pendahuluan

Sebuah program yang baik adalah program yang membagi permasalahan utama menjadi bagian-bagian kecil dimana setiap bagian kecil ditangani oleh sebuah subprogram, cara ini disebut dengan *modular programming* (pemrograman terbagi/terpecah). Cara ini termasuk pemrograman terstruktur dan sangat didukung oleh bahasa Pascal. Untuk itu, Pascal telah menyediakan dua jenis subprogram, yaitu *procedure* dan *function* (prosedur dan fungsi).

Dengan *modular programming*, program lebih mudah dibaca dan dimengerti. Selain itu, pembenahan program dan penelusuran jalannya program (*debugging*) menjadi lebih mudah sebab dapat langsung diketahui subprogram mana yang berjalan tidak sesuai dengan yang diharapkan.

## 2.2. P r o s e d u r

Prosedur adalah subprogram yang menerima masukan tetapi tidak mempunyai keluaran secara langsung. Cara mendeklarasikan sebuah prosedur adalah sebagai berikut :

```
procedure A;            { nama prosedur adalah A }
begin
  { statement }
end;
```

Pendeklarasian prosedur di atas adalah untuk prosedur yang tidak memerlukan parameter. Parameter adalah data masukan untuk subprogram yang nantinya akan diproses lebih lanjut dalam subprogram tersebut. Dalam Pascal, dikenal dua macam parameter yaitu :

1. parameter nilai (*value parameter*), dan
2. parameter referensi (*reference parameter*).

Cara mendeklarasikan parameter tersebut adalah sebagai berikut :

```
procedure B(X : integer; var Y : integer);
begin
  { statement }
end;
```

Pada deklarasi prosedur di atas, parameter X adalah parameter nilai sedang parameter Y adalah parameter referensi. Jadi, pendeklarasian parameter referensi didahului oleh *reserved word* **var**. Parameter referensi ini nantinya dapat dijadikan sebagai variabel keluaran dari prosedur.

Untuk lebih memahami penggunaan prosedur dalam Pascal, perhatikan contoh program di bawah ini :

```
program Prosedur;

uses CRT;

var
  Bil_1, Bil_2, Hasil : integer;

procedure Awal;
begin
  Writeln('Latihan Pascal 2 : Prosedur dan Fungsi');
  Writeln('-------------------------------------');
  Writeln;
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
end;

procedure Baca_Data;
begin
  Write('Masukkan bilangan pertama : ');
  Readln(Bil_1);
  Write('Masukkan bilangan kedua   : ');
  Readln(Bil_2);
  Writeln;
end;

procedure Kali(A,B : integer);
var
  I : integer;
begin
  Hasil := 0;
  for I := 1 to B do Hasil := Hasil + A;
end;

procedure Kalikan(A,B : integer; var C : integer);
var
  I : integer;
begin
  C := 0;
  for I := 1 to B do C := C + A;
end;

begin
  ClrScr;
  Awal;
  Baca_Data;
  Kali(Bil_1, Bil_2);
  Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Hasil:5);
  Kalikan(Bil_1, Bil_2, Hasil);
```

```
  Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Hasil:5);
  Writeln;
  Write('Tekan Enter...');
  Readln;
end.
```

Perhatikan program di atas. Dua prosedur terakhir memiliki kemiripan, bedanya hanya pada jumlah parameter dan variabel hasil perkaliannya. Untuk lebih jelas, jalankan program dan perhatikan apa yang dilakukan oleh dua prosedur tersebut maka akan nampak perbedaan keduanya.

## 2.3. F u n g s i

Fungsi adalah subprogram yang menerima masukan dan mempunyai keluaran secara langsung. Cara mendeklarasikan sebuah fungsi adalah sebagai berikut :

```
function A : integer;      { nama fungsi adalah A dengan }
begin                      { tipe data keluaran adalah integer }
  { statement }
  A := 3;                  { nilai yang dikeluarkan fungsi }
end;
```

Sebagaimana dalam prosedur, fungsi juga dapat diberikan parameter. Cara mendeklarasikan fungsi dengan parameter juga tidak jauh berbeda dengan pendeklarasian parameter pada prosedur.

```
function B(X : integer) : integer;
begin
  { statement }
  B := X * 2;
end;
```

Perbedaan utama antara prosedur dan fungsi adalah dalam menghasilkan keluaran. Walaupun prosedur bisa menghasilkan nilai keluaran, tetapi nilai tersebut tidak dapat diambil secara langsung, melainkan harus diambil melalui parameter referensi. Sedangkan keluaran dari fungsi dapat diambil langsung dari fungsi tersebut. Untuk lebih memahami perbedaan prosedur dan fungsi, perhatikan contoh berikut ini :

```
program Fungsi;

uses CRT;

var
  Bil_1, Bil_2, Hasil : integer;
```

```
procedure Awal;
begin
  Writeln('Latihan Pascal 2 : Prosedur dan Fungsi');
  Writeln('-------------------------------------');
  Writeln;
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
end;

procedure Baca_Data;
begin
  Write('Masukkan bilangan pertama : ');
  Readln(Bil_1);
  Write('Masukkan bilangan kedua   : ');
  Readln(Bil_2);
  Writeln;
end;

function Kali(A,B : integer) : integer;
var
  I,J : integer;
begin
  J := 0;
  for I := 1 to B do J := J + A;
  Kali := J;
end;

procedure Kalikan(A,B : integer; var C : integer);
var
  I : integer;
begin
  C := 0;
  for I := 1 to B do C := C + A;
end;

begin
  ClrScr;
  Awal;
  Baca_Data;
  Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Kali(Bil_1,Bil_2):5);
  Kalikan(Bil_1, Bil_2, Hasil);
  Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Hasil:5);
  Writeln;
  Write('Tekan Enter...');
  Readln;
end.
```

Perhatikan program di atas. Prosedur Kalikan dan fugsi Kali mempunyai keluaran yang sama, tetapi cara mengambil keluarannya berbeda. Perhatikan dan jelaskan apa yang terjadi jika baris keempat dalam program utama yang semula perintah :

```
Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Kali(Bil_1,Bil_2):5);
```

diubah menjadi :

```
Writeln(Bil_1:3,' x ',Bil_2:3,' = ',Kalikan(Bil_1,Bil_2,Hasil):5);
```

## 2.4. Rekursi

Dalam Pascal, ada satu kelebihan dalam cara pemanggilan subprogram. Pascal mengijinkan pemanggilan suatu subprogram dari dalam subprogram itu sendiri. Tidak semua bahasa pemrograman mengijinkan cara pemanggilan subprogram seperti itu karena akan banyak memakan memori. Untuk lebih jelasnya perhatikan potongan program di bawah ini :

```
procedure Z;
begin
  { statement }
  Z;
end;
```

Pada baris terakhir prosedur Z di atas, terdapat pemanggilan kembali terhadap prosedur Z, sehingga prosedur di atas tidak akan pernah selesai dijalankan sebab begitu sampai pada baris terakhir dari prosedur, program akan kembali lagi ke awal prosedur. Yang terjadi adalah semacam perulangan tanpa perintah perulangan Pascal, dan perulangan dengan cara ini disebut dengan rekursi. Rekursi berlaku terhadap semua subprogram dalam Pascal, yaitu prosedur dan fungsi.

Dengan adanya rekursi ini, banyak algoritma komputer menjadi lebih mudah dibuat programnya. Berikut ini adalah program menghitung suku banyak Legendre, salah satu contoh perhitungan yang dapat diselesaikan dengan menggunakan rekursi :

```
program Rekursi;

uses CRT;

var
  Jum_Suku, I : integer;
  Bil_X       : real;

function Legendre(X : real; N : integer) : real;
var
  Suku_1, Suku_2 : real;
begin
  if N = 0 then
    Legendre := 1
  else if N = 1 then
    Legendre := X
  else
    begin
      Suku_1 := ((2*N - 1) * (X * Legendre(X, N-1))) / N;
      Suku_2 := ((N-1) * Legendre(X, N-2)) / N;
      Legendre := Suku_1 + Suku_2;
    end;
end;
```

```
procedure Awal;
begin
  Writeln('Latihan Pascal 2 : Prosedur dan Fungsi');
  Writeln('-------------------------------------');
  Writeln;
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
end;

procedure Baca_Data;
begin
  Writeln('Menghitung Suku Banyak Legendre');
  Writeln;
  Write('Sampai suku ke   : ');
  Readln(Jum_Suku);
  Write('Masukkan nilai X : ');
  Readln(Bil_X);
  Writeln;
end;

begin
  ClrScr;
  Awal;
  Baca_Data;

  for I := 0 to Jum_Suku do
  begin
    Writeln('Suku ke-',I:2,', Nilainya = ',Legendre(Bil_X, I):8:3);
  end;

  Writeln;
  Write('Tekan Enter...');
  Readln;
end.
```

Untuk lebih jelas memahami program, jalankan program dengan F7. Perhatikan pula apa yang dilakukan oleh fungsi Legendre. Amati perubahan variabel-variabel yang terlibat dalam fungsi.

## 2.5. Latihan

1. Apa guna parameter dalam subprogram ?
2. Kapan kita menggunakan fungsi dan kapan pula menggunakan prosedur?
3. Sebutkan dan jelaskan kelemahan rekursi !
4. Buatlah program Rekursi di atas tanpa menggunakan rekursi kemudian bandingkan antara yang menggunakan rekursi dan yang tidak. Jelaskan perbedaan dan persamaannya !
5. Buatlah program untuk menghitung perkalian dua bilangan kompleks ! Gunakan prosedur atau fungsi !

## LATIHAN III
# ARRAY DAN RECORD

### 3.1. Pendahuluan

Dalam bahasa Pascal, secara garis besar dikenal dua macam tipe data yaitu tipe data sederhana (*primitive type*) dan tipe data kompleks (*complex type*). Contoh tipe data sederhana adalah tipe numerik (integer dan real), tipe data karakter, tipe data *boolean* dan tipe data enumerasi. Contoh tipe data kompleks adalah *string*, *array* (larik), *record* dan *object*.

Tipe data sederhana adalah tipe data yang hanya mampu menyimpan satu nilai tiap satu variabelnya. Sebaliknya tipe data kompleks adalah tipe data yang mampu menyimpan lebih dari satu nilai dalam tiap satu variabelnya. Dalam latihan ini hanya akan dibahas dua tipe data kompleks yaitu *array* dan *record*.

### 3.2. A r r a y

Array adalah tipe data kompleks yang elemen-elemennya mempunyai tipe data yang sama. Jumlah elemen array bersifat tetap dan tidak bisa ditambah atau dikurangi setelah pendeklarasiannya. Tiap elemen mempunyai nomer indeks sendiri dan pengaksesan terhadap elemen array dilakukan dengan menunjukkan nomer indeks dari elemen yang akan diakses.

Cara pendeklarasian suatu variabel bertipe array adalah sebagai berikut :

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

Pada potongan program di atas, maksudnya adalah sebagai berikut : variabel A berupa array dari integer dengan jumlah elemen sebanyak 10, nomer indeks terkecil adalah 1 dan nomer indeks terbesar adalah 10. Untuk mengakses elemen dari variabel A dapat dilakukan dengan menunjukkan nomer indeks elemen A seperti ini : `A[nomer_indeks]` contoh : `A[1] := 10;`

Untuk lebih memahami penggunaan array dalam program, perhatikan contoh program di bawah ini :

```
program Fibonacci;
uses CRT;
```

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

procedure Awal;
begin
  Writeln('Praktikum DKP III : Array dan Record');
  Writeln('-----------------------------------');
  Writeln;
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
end;

procedure Fibo;
begin
  for I := 1 to 10 do
  begin
    if I < 3 then
      Data[I] := I - 1
    else
      Data[I] := Data[I-1] + Data[I-2];
  end;

  Writeln('Deret Fibonacci suku ke-1 hingga suku ke-10 :');
  for I := 1 to 10 do Write(Data[I]:3);
  Writeln;
end;

begin
  ClrScr;
  Awal;
  Fibo;
  Writeln;
  Write('Tekan Enter...');
  Readln;
end.
```

Perhatikan program di atas, terutama cara mengakses variabel array pada prosedur Fibo. Untuk lebih memahami jalannya program, jalankan program dengan F7 dan perhatikan perubahan elemen-elemen variabel Data yang berupa array.

Jika program di atas telah dimengerti, buatlah program menghitung deret Fibonacci tetapi tidak menggunakan variabel array. Kemudian bandingkan dan perhatikan perbedaan kedua program tersebut.

Array yang digunakan pada program di atas adalah array berdimensi tunggal atau array berdimensi satu. Dengan demikian, dapat pula dideklarasikan variabel array dengan dimensi lebih dari satu atau array berdimensi banyak. Berikut adalah cara mendeklarasikan array berdimensi dua :

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

Antara dimensi satu dengan dimensi lainnya dipisahkan oleh tanda koma (,), demikian juga untuk mendeklarasikan array berdimensi lebih dari dua.

Cara mengakses elemen array juga tidak jauh berbeda dengan cara mengakses elemen array berdimensi satu, yaitu menggunakan nomer indeksnya. Contohnya sebagai berikut : `A[2,3] := 10;` artinya elemen yang terletak pada nomer 2 dimensi pertama dan nomer 3 dimensi kedua diisi dengan nilai 10.

Array berdimensi dua ini banyak digunakan dalam perhitungan matrik, oleh sebab itu array berdimensi dua disebut juga dengan array matrik. Perhatikan contoh program berikut ini :

```
program Jumlah_Matrik;

uses CRT;

const
  Orde = 3;

type
  Matrik = array[1..orde,1..orde] of integer;

var
  M1, M2, H : matrik;
  I, J      : integer;

procedure Awal;
begin
  Writeln('Latihan Pascal 3  : Array dan Record');
  Writeln('-----------------------------------');
  Writeln;
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
end;

procedure JumlahMatrik(var Mat1, Mat2, MatHasil : matrik);
begin
  for I := 1 to orde do
    for J := 1 to orde do
      MatHasil[I,J] := Mat1[I,J] + Mat2[I,J];
end;

procedure BacaData(var Mat : matrik);
begin
  for I := 1 to orde do
    for J := 1 to orde do
    begin
      Write('Nilai[',I,',',J,'] = ');
      Readln(Mat[I,J]);
    end;
end;
```

```
procedure TulisMatrik(var Mat : matrik);
begin
  for I := 1 to orde do
  begin
    for J := 1 to orde do
    begin
      Write(Mat[I,J]:5);
    end;
    Writeln;
  end;
end;

begin
  ClrScr;
  Awal;
  Writeln('Isi matrik pertama :');
  BacaData(M1);
  Writeln;
  Writeln('Isi matrik kedua :');
  BacaData(M2);
  Writeln;

  JumlahMatrik(M1, M2, H);

  Writeln('Penjumlahan matrik pertama dan kedua :');
  TulisMatrik(H);
  Writeln;
  Write('Tekan Enter...');
  Readln;
end.
```

Perhatikan program di atas. Terutama cara mengakses isi array dua dimensi secara berurutan seperti pada tiga prosedur terakhir.

### 3.3. R e c o r d

Record adalah tipe data kompleks yang elemen-elemennya boleh mempunyai tipe data yang berbeda. Record lebih kompleks daripada array karena record merupakan kumpulan beberapa variabel dengan tipe data yang berbeda. Berbeda dengan array yang tiap elemennya ditandai dengan nomer indeks maka record ditandai dengan nama variabel anggotanya. Cara mengakses elemen dari record dilakukan dengan menyebutkan nama variabel anggota setelah menyebutkan nama record yang akan diakses. Di antara nama record dan nama variabel anggota dipisahkan tanda titik (**.**).

Cara pendeklarasian record adalah sebagai berikut :

```
var
  B : record
        X : integer;
        Y : real;
      end;
```

Pada pendeklarasian di atas, maksudnya adalah sebagai berikut : variabel B berupa record dengan dua elemen yaitu X bertipe integer dan Y bertipe real. Untuk mengakses elemen dari variabel B seperti berikut : `B.nama_variabel`

contoh : `B.X := 10;`

Untuk lebih memahami penggunaan record dalam program, perhatikan contoh berikut ini :

```
program Jumlah_Kompleks;

uses CRT;

Type
  Kompleks = record
               bil_real : integer;
               bil_imaj : integer;
             end;

var
  K1, K2, H : kompleks;

procedure Awal;
begin
  Writeln(' Latihan Pascal 3 : Array dan Record');
  Writeln('-----------------------------------');
  Writeln;
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
end;

procedure JumlahKompleks(var Komp1, Komp2, KompHasil : kompleks);
begin
  KompHasil.bil_real := Komp1.bil_real + Komp2.bil_real;
  KompHasil.bil_imaj := Komp1.bil_imaj + Komp2.bil_imaj;
end;

procedure BacaData(var Komp : kompleks);
begin
  Write('Bilangan real : ');
  Readln(Komp.bil_real);
  Write('Bilangan imajiner : ');
  Readln(Komp.bil_imaj);
end;

procedure TulisKompleks(var Komp : kompleks);
begin
  Write('(',Komp.bil_real:3,' + ',Komp.bil_imaj:3,'i)');
end;

begin
  ClrScr;
  Awal;
  Writeln('Isi bilangan kompleks pertama :');
  BacaData(K1);
  Writeln;
  Writeln('Isi bilangan kompleks kedua :');
```

```
  BacaData(K2);
  Writeln;

  JumlahKompleks(K1, K2, H);

  Writeln('Penjumlahan bilangan kompleks pertama dan kedua :');
  TulisKompleks(K1);
  Write(' + ');
  TulisKompleks(K2);
  Write(' = ');
  TulisKompleks(H);
  Writeln;
  Writeln;
  Write('Tekan Enter...');
  Readln;
end.
```

Perhatikan program di atas. Untuk lebih jelasnya, jalankan program dengan F7 sehingga akan terlihat urutan jalannya program. Perhatikan pula bagaimana cara mengakses elemen record seperti pada prosedur JumlahKompleks.

### 3.4. Latihan

1. Buatlah program untuk mengurutkan sekumpulan data !
2. Buatlah program menghitung perkalian matriks !
3. Buatlah program menghitung perkalian bilangan kompleks !
4. Jelaskan guna *reserved word* **`with`** dalam Pascal !
5. Jelaskan perbedaan dan persamaan antara array dan record !

# LATIHAN IV
# G R A F I K

## 4.1. Pendahuluan

Layar komputer selain dapat beroperasi pada mode teks, juga dapat beroperasi pada mode grafik. Dengan adanya kemampuan ini, banyak hal yang semula tidak dapat dilakukan pada mode teks dapat dilakukan pada mode grafik, seperti membuat atau menampilkan gambar di layar komputer. Selain itu, mode grafik dapat digunakan untuk memvisualisasikan grafik persamaan matematika.

Karena mode grafik berbeda dengan mode teks, maka pemrogramannya pun berbeda pula. FreePascal telah menyediakan prosedur dan fungsi khusus untuk pemrograman pada mode grafik. Latihan keempat ini akan mempelajari pemrograman mode grafik dengan FreePascal.

## 4.2. Mengaktifkan Mode Grafik

Untuk mengaktifkan mode grafik, digunakan prosedur standar Pascal :

`InitGraph`(**var** GraphDriver: integer; **var** GraphMode: integer; DriverPath: **string**);

Pada prosedur di atas, diperlukan tiga buah parameter :

1. `GraphDriver` : definisi tingkat kedalaman warna monitor,
2. `GraphMode` : definisi resolusi monitor,
3. `DriverPath` : lokasi file *driver* yang digunakan, jika menggunakan file khusus, biasanya tidak perlu sehingga bisa dikosongkan.

## 4.3. Prosedur dan Fungsi Grafik Standar Pascal

Untuk penanganan mode grafik, FreePascal telah menyediakan banyak prosedur dan fungsi dalam satu unit khusus untuk pemrograman grafik. Lebih jelasnya, perhatikan program berikut :

```
program Grafik1;

uses Graph;

var
  GDriver, GMode : integer;
  C, I, X, Y     : integer;
  SX, SY, SC     : string;

begin
  GDriver := d8bit;   // kedalaman warna 8 bit (256 warna)
  GMode := m640x480;  // resolusi monitor 640 lebar x 480 tinggi
  InitGraph(GDriver,GMode,'');
```

```
  Randomize;
  C := GetMaxColor;
  X := GetMaxX;
  Y := GetMaxY;
  Str(C+1,SC);
  Str(X+1,SX);
  Str(Y+1,SY);

  for I := 1 to 100 do
  begin
    PutPixel(random(X),random(Y),random(C));
    SetColor(random(C));
    Line(random(X),random(Y),random(X),random(Y));
    Circle(random(X),random(Y),random(50));
  end;

  Bar(2,2,637,10);
  SetColor(0);
  OutTextXY(5,3,Latihan 4 : Grafik');
  OutTextXY(500,3,'Nama : ---------');
  Bar(2,468,637,477);
  SetColor(1);
  OutTextXY(380,469,'Mode Grafik VGA'+SX+'x'+SY+' '+SC+' warna');
  SetColor(4);
  OutTextXY(5,469,'Tekan Enter...');
  SetColor(C);
  Rectangle(0,0,X,Y);
  Readln;
  CloseGraph;
end.
```

Perhatikan jalannya program dengan cara menjalankannya perbaris (*step over*), gunakan tombol F8. Dengan cara ini dapat dilihat apa yang dilakukan oleh tiap baris program di atas.

## 4.4. Aplikasi Grafik

Setelah mengetahui cara pemrograman grafik, hal selanjutnya yang perlu dipelajari adalah cara mengaplikasikannya untuk menyelesaikan suatu permasalahan. Percobaan kali ini akan membahas pemanfaatan mode grafik untuk menampilkan bentuk grafik suatu persamaan matematika. Perhatikan program berikut ini :

```
program Grafik2;

uses Graph;

var
  GDriver, GMode : integer;
  A, B, I        : integer;
  MC, MX, MY     : integer;
  TX, TY, JX, JY : integer;
  K, X1, X2, Y   : real;
  SX, SY, SC     : string;
```

```
function F(X : real): real;
begin
  F := sin(2*PI*X)/X;
end;

begin
  GDriver := d8bit;
  GMode := m640x480;
  InitGraph(GDriver, GMode, 'A:\');

  MC := GetMaxColor;
  MX := GetMaxX;
  MY := GetMaxY;
  Str(MC+1,SC);
  Str(MX+1,SX);
  Str(MY+1,SY);

  Bar(2,2,637,10);
  SetColor(0);
  OutTextXY(5,3,'Latihan 4 : Aplikasi Grafik');
  OutTextXY(500,3,'Nama : ----------');
  Bar(2,468,637,477);
  SetColor(1);
  OutTextXY(370,469,'Mode Grafik VGA '+SX+'x'+SY+' '+SC+' warna');
  SetColor(4);
  OutTextXY(5,469,'Tekan Enter...');
  SetColor(MC);
  Rectangle(0,0,MX,MY);
  Rectangle(2,12,637,466);
  SetViewPort(3,13,636,465,true);

  TX := 320;
  TY := 340;
  JX := 50;
  JY := 50;
  X1 := -25;
  X2 := 25;
  K  := 0.001;
  SetColor(9);
  Line(TX,0,TX,MY);
  Line(0,TY,MX,TY);
  SetLineStyle(1,0,1);
  SetColor(8);

  for I := 1 to 25 do
  begin
    A := TX + I*JX;
    B := TX - I*JX;
    Line(A,0,A,MY);
    Line(B,0,B,MY);
    A := TY - I*JY;
    B := TY + I*JY;
    Line(0,A,MX,A);
    Line(0,B,MX,B);
  end;

  repeat
    X1 := X1 + K;
    Y := F(X1);
    A := round(X1 * JX) + TX;
    B := TY - round(Y * JY);
```

```
    PutPixel(A,B,14);
  until X1 > X2;

  Readln;
  CloseGraph;
end.
```

Jalankan program di atas dengan menekan tombol F8. Untuk lebih jelas memahami program lakukan beberapa hal berikut :

- Ubah nilai variabel TX menjadi …..…..., kemudian jalankan kembali program di atas. Ubah lagi variabel TY menjadi ……..., kemudian jalankan lagi programnya.
- Ubah nilai variabel JX menjadi ……., kemudian jalankan kembali program di atas. Ubah lagi variabel JY menjadi ……., kemudian jalankan lagi programnya.
- Ubah nilai variabel X1 menjadi ……, kemudian jalankan kembali program di atas. Ubah variabel X2 menjadi ……, kemudian jalankan lagi programnya.
- Ubah nilai variabel K menjadi ……., kemudian jalankan lagi program di atas. Ubah lagi variabel K menjadi ……., kemudian jalankan lagi programnya.

Dengan melakukan beberapa hal di atas, dapat ditarik kesimpulan mengenai manfaat masing-masing variabel yang digunakan dalam program.

### 4.5. Latihan

1. Jelaskan (berikut parameternya) dari fungsi dan prosedur grafik Pascal di bawah ini:
   a. `GetPixel`
   b. `ClearDevice`
   c. `Bar3D`
   d. `SetLineStyle`
   e. `SetTextStyle`
2. Buatlah flowchart dari program Grafik2, kemudian jelaskanlah jalan programnya !
3. Jelaskan manfaat penggunaan fungsi `round` dalam perulangan `repeat...until` pada program Grafik2 !
4. Jelaskan manfaat dari masing-masing variabel yang telah Anda amati pada saat menjalankan program Grafik2 di atas !
5. Dengan menggunakan program Grafik2, ubahlah persamaan grafiknya menjadi : `f(x) = exp(x)`. Jelaskan apa yang terjadi pada program !

# LATIHAN V
# ANALISA NUMERIK

## 5.1. Pendahuluan

Setelah mempelajari pemrograman menggunakan bahasa Pascal maka pada latihan terakhir ini akan dipelajari tentang aplikasi pemrograman untuk memecahkan permasalahan dengan metode numerik. Metode numerik adalah teknik yang digunakan untuk memformulasikan masalah matematis sehingga dapat dipecahkan dengan operasi perhitungan. Metode numerik banyak berguna dalam memecahkan masalah-masalah teknik rekayasa yang tidak mungkin dipecahkan secara analitis.

Dalam latihan ini hanya dibahas dua masalah dasar dalam metode numerik, yaitu penentuan akar-akar persamaan dan menyelesaikan sistem persamaan. Selain itu, pembahasan akan lebih ditekankan pada materi pemrogramannya daripada teori numeriknya. Lebih lanjut tentang teori-teori metode numerik dapat dipelajari dalam mata pelajaran Matematika.

## 5.2. Menentukan Akar-Akar Persamaan

Suatu persamaan matematika dapat ditentukan akar persamaannya dengan beberapa metode. Secara garis besar, ada dua metode yaitu metode pengurung dan metode terbuka. Salah satu metode pengurung adalah metode grafik, metode ini sudah dijelaskan pada latihan keempat tentang aplikasi grafik. Metode ini mengamati secara langsung posisi akar pada grafik persamaannya. Selain metode grafik, ada beberapa macam lagi metode pengurung seperti metode inkremental, metode bagi dua dan metode posisi palsu. Berikut ini adalah program untuk menentukan akar-akar persamaan dengan metode inkremental :

```
program Akar_Persamaan;

uses CRT;

var
  MaxGalat, AkarPers,
  BatasAtas, BatasBawah : real;
  MaxIterasi            : integer;


function F(X : real) : real;
begin
  F := X*X*X - 6*X - 7;
end;
```

```
procedure Awal;
begin
  Writeln('Latihan 5 : Analisa Numerik');
  Writeln('--------------------------');
  Writeln;
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
  Writeln('Program Untuk Menghitung Akar Persamaan');
  Writeln;
end;

procedure BacaData;
begin
  Write('Masukkan galat maksimum   : ');
  Readln(MaxGalat);
  Write('Masukkan maksimum iterasi : ');
  Readln(MaxIterasi);
  Write('Masukkan perkiraan awal   : ');
  Readln(BatasBawah);
  Writeln;
end;

procedure Inkremental;
var
  Iterasi    : integer;
  Kenaikan,
  Galat, Uji : real;

begin
  Iterasi := 0;
  Kenaikan := 1;
  Writeln(' Iterasi   Batas Bawah    Akar Persamaan    Galat');
  Writeln('-----------------------------------------------');

  repeat
    Iterasi := Iterasi + 1;
    BatasAtas := BatasBawah + Kenaikan;
    AkarPers := BatasAtas;

    Uji := F(BatasAtas) * F(BatasBawah);
    Galat := abs((AkarPers - BatasBawah)/AkarPers);

    if Uji > 0 then BatasBawah := BatasAtas
      else Kenaikan := Kenaikan/2;

    Writeln(Iterasi:5, BatasBawah:17:5, AkarPers:15:5, Galat:13:5);
  until (Kenaikan <= MaxGalat) or (Iterasi >= MaxIterasi);

  Writeln('-----------------------------------------------');
  Writeln;
end;


procedure TampilHasil;
begin
  Writeln('Akar persamaan dari fungsi adalah ',AkarPers:8:5);
end;
```

```
begin
  ClrScr;
  Awal;

  BacaData;
  Inkremental;
  TampilHasil;

  Writeln;
  Write('Tekan Enter...');
  Readln;
end.
```

Jalankan program di atas dengan menekan tombol F7, perhatikan jalannya program dan pelajari algoritmanya. Jika algoritma program telah dimengerti, temukan kelemahan dan kelebihannya dan perbaikilah kelemahan tersebut.

## 5.3. Menyelesaikan Sistem Persamaan Linier

Dalam teknik-teknik rekayasa seringkali dijumpai suatu permasalahan yang lebih mudah menyelesaikannya dengan mengubah permasalahan menjadi persamaan matematika. Untuk persamaan tunggal dapat diselesaikan dengan menggunakan metode penentuan akar persamaan, sedang untuk persamaan yang lebih dari satu maka diperlukan metode lain untuk menyelesaikannya.

Ada beberapa metode numerik yang seringkali digunakan untuk menyelesaikan suatu sistem persamaan, antara lain dengan metode eliminasi matrik, metode grafik dan metode eliminasi Gauss yang mempunyai beberapa macam jenis. Metode grafik telah dibahas pada latihan sebelumnya, tetapi untuk menyelesaikan suatu sistem persamaan, semua grafik persamaan harus ditampilkan bersamaan, karenanya contoh program dalam latihan keempat harus diperbaiki agar dapat menampilkan beberapa persamaan sekaligus. Berikut ini adalah contoh program menggunakan metode eliminasi matrik :

```
Program Eliminasi_Matrik;

uses CRT;

const
  JmlPers = 3;

type
  Matrik = array[1..JmlPers+1, 1..JmlPers+1] of real;

var
  Koefs  : matrik;
```

```
procedure Identitas;
begin
  Writeln('Latihan 5 : Analisa Numerik');
  Writeln('--------------------------');
  Writeln;
  Writeln('Nama : _____________________');
  Writeln('NIM  : __________');
  Writeln;
end;

procedure Judul;
begin
  Writeln('Program Penyelesaian 3 Persamaan Linier');
  Writeln;
  Writeln('Bentuk persamaan :  a1 x + b1 y + c1 z = k1');
  Writeln('                    a2 x + b2 y + c2 z = k2');
  Writeln('                    a3 x + b3 y + c3 z = k3');
  Writeln;
end;

procedure BacaData;
var
  I, J : integer;
begin
  for I := 1 to JmlPers do
  begin
    for J := 1 to JmlPers + 1 do
    begin
      if J = JmlPers + 1 then
      begin
        Write('Masukkan konstanta k',I,' : ');
        Readln(Koefs[I,J]);
      end
      else
        begin
          Write('Masukkan nilai ',chr(96+J),I,' : ');
          Readln(Koefs[I,J]);
        end;
    end;
    Writeln;
  end;
end;

function Det3x3(var Mat : matrik) : real;
var
  Det3, H    : real;
  I, J, K, L : integer;
  Det2       : array[1..4] of real;
begin
  K := 0;
  Det3 := 0;

  for L := 1 to 4 do Det2[L] := 0;
  for I := 1 to 3 do
  begin
    for L := 2 to 3 do
    begin
      for J := 1 to 3 do
      begin
        if I <> J then
```

```
        begin
          K := K + 1;
          Det2[K] := Mat[L,J];
        end;
      end;
    end;

    H := Mat[1,I];
    if I mod 2 = 0 then H := -H;
    Det3 := Det3 + (Det2[1]*Det2[4] - Det2[2]*Det2[3]) * H;
    for L := 1 to 4 do Det2[L] := 0;
    K := 0;
  end;

  Det3x3 := Det3;
end;

procedure EliminasiMatrik;
var
  MatElim : matrik;
  I, J    : integer;
  A, B    : real;
begin
  MatElim := Koefs;

  for J := 1 to JmlPers do
  begin
    for I := 1 to JmlPers do
      MatElim[I,J] := Koefs[I,JmlPers+1];

    A := Det3x3(MatElim);
    B := Det3x3(Koefs);
    Koefs[JmlPers+1,J] := A/B;
    MatElim := Koefs;
  end;
end;

procedure TampilkanHasil;
var
  I : integer;
begin
  ClrScr;
  Identitas;
  Writeln('Program Penyelesaian 3 Persamaan Linier');
  Writeln;
  Writeln('Bentuk persamaan :  ');

  for I := 1 to JmlPers do
  begin
    Write(Koefs[I,1]:5:2,'x + ',Koefs[I,2]:5:2,'y + ');
    Writeln(Koefs[I,3]:5:2,'z = ',Koefs[I,4]:5:2);
  end;

  Writeln;
  Writeln('Penyelesaian persamaan :');

  for I := 1 to JmlPers do
    Writeln(chr(119+I):5,' = ',Koefs[JmlPers+1,I]:5:2);
end;
```

```
begin
  ClrScr;
  Identitas;
  Judul;

  BacaData;
  EliminasiMatrik;
  TampilkanHasil;

  Writeln;
  Write('Tekan Enter...');
  Readln;
end.
```

Jalankan program di atas dengan tombol F7. Perhatikan terutama pada prosedur EliminasiMatrik sebab prosedur itulah inti dari program di atas. Fungsi Det3x3 yang menghasilkan nilai determinan dari matrik orde 3x3 juga sangat penting untuk dimengerti. Diharapkan setelah memahami program di atas, bisa lebih mudah untuk mempelajari algoritma penyelesaian sistem persamaan linier lainnya, seperti metode eliminasi Gauss, eliminasi Gauss-Jordan dan lain sebagainya.

## 5.4. Latihan

1. Buatlah program untuk menghitung perkalian matrik berukuran 3 x 3 !
2. Buatlah program untuk menghitung gaya momentum dua benda yang saling bertumbukan !