Modul 1:

Pemrograman FORTRAN dan Analisis Galat (*errors*) (Disertai Translasi Turbo Pascal ke FORTRAN)

Materi pokok yang berhubungan dengan masalah-masalah dan atau problem-problem matematika terapan (***advanced engineering mathematics***) dan pemodelan (***mathematical modelling***) selalu berkaitan erat dengan metode dan atau analisis numerik, karena keduanya selalu diharapkan memiliki atau memberikan solusi yang praktis, efisien dan dengan akurasi (ketelitian) yang memadai.

Penggunaan metode-metode numerik tidak dapat dipisahkan dari masalah-masalah penggunaan komputer, terutama 'galat' ataupun 'sesatan' (*error*) yang ditimbulkan oleh penggunaan CPU komputer (prosesor dan koprosesor), sebagai piranti keras komputasi numerik. Komputer-komputer yang digunakan pada saat ini, semuanya sudah dapat digolongkan dalam 'komputer digital' yang kecepatannya jauh lebih tinggi dibandingkan 'komputer analog', di samping itu juga penggunaan komputer digital sangat tepat dan sesuai untuk metode perhitungan pendekatan secara numerik (bukan analitis).

Pada bagian-bagian awal dari modul ini akan dibahas terlebih dahulu secara ringkas dan cepat tentang 'bahasa pemrograman' (*programming language*). Karena kepopulerannya yang mendunia, juga karena kemudahan dan ketersediaan pustaka (*library*) di berbagai literatur dan lembaga pendidikan terkemuka di dunia, maka **Bahasa FORTRAN** (singkatan dari FORmula TRANslation) lebih diutamakan dalam penyajian modul-modul metode numerik dalam buku ini.

Selanjutnya, akan dibahas pula hal-hal yang berhubungan dengan analisis dan studi matematis tentang galat dan konvergensi.

Analisis galat merupakan hal yang terpenting dalam analisis numerik, mengingat metode-metode numerik yang digunakan semuanya memberikan jawaban yang bersifat ‘pendekatan’ (*aproksimatif*) terhadap jawaban sebenarnya (*solusi eksak*).

Untuk memudahkan dan juga mengingatkan para pembaca dalam melakukan pemrograman metode numerik menggunakan Bahasa FORTRAN, dalam modul ini juga disertakan pembahasan ringkas tentang FORTRAN (terutama versi ANSI 77 dan sedikit tentang ANSI 90 dan 95), disamping beberapa contoh terjemahan Bahasa Turbo PASCAL ke dalam FORTRAN. Keterlibatan mahasiswa akan penggunaan Bahasa Turbo PASCAL di Jurusan Teknik Gas dan Petrokimia sudah relatif lama, namun ternyata dirasakan sulit berkembang karena masalah-masalah ketersediaan pustaka dan juga masalah globlalisasi pendidikan keteknikan yang sudah membudaya dengan Bahasa FORTRAN.

A. Pemrograman dan Bahasa Pemrograman

Bagi para pemrogram (*programmers*), baik dalam BASIC, Turbo PASCAL, FORTRAN, C, atau lainnya, modul ini sebenarnya samasekali belum lengkap. Namun, beberapa materi dalam modul ini dapat digunakan sebagai ‘alat kelengkapan’ dan ‘pengingat praktis’ bagi peserta ajar yang lebih melibatkan-diri pada analisis dan metode-metode numerik.

Selain itu juga, dalam penyajian modul ini diharapkan para pembaca telah sedikit memahami beberapa bahasa pemrograman, sedemikian rupa sehingga yang bersangkutan masih dapat mengerti logika pembuatan ‘program terstruktur’ (*structured programming*) sebagai ‘inti’ dari implementasi metode-metode komputasi numerik, meskipun Bahasa FORTRAN masih belum dikuasainya.

Sebenarnya bahasa-bahasa pemrograman yang umum seperti BASIC, FORTRAN, Turbo PASCAL, dan C telah lama digunakan di dalam analisis dan komputasi numerik, namun tampaknya

kesepakatan akan penggunaan ‘bahasa pemrograman yang baku’ masih belum tercapai. Namun, disisi lain, dengan semakin luasnya penggunaan ‘platform’ OS (*operating system* atau sistem operasi) berbasis UNIX (termasuk CENIX dan Sun Solaris) di Eropa Utara dan Amerika Serikat pada dekade tahun 1980-an untuk komputer-komputer midi dan super, sebenarnya kesepakatan ‘secara tidak langsung’ telah terbangun, yaitu: penggunaan Bahasa C sebagai platform dasar untuk bahasa pemrograman.

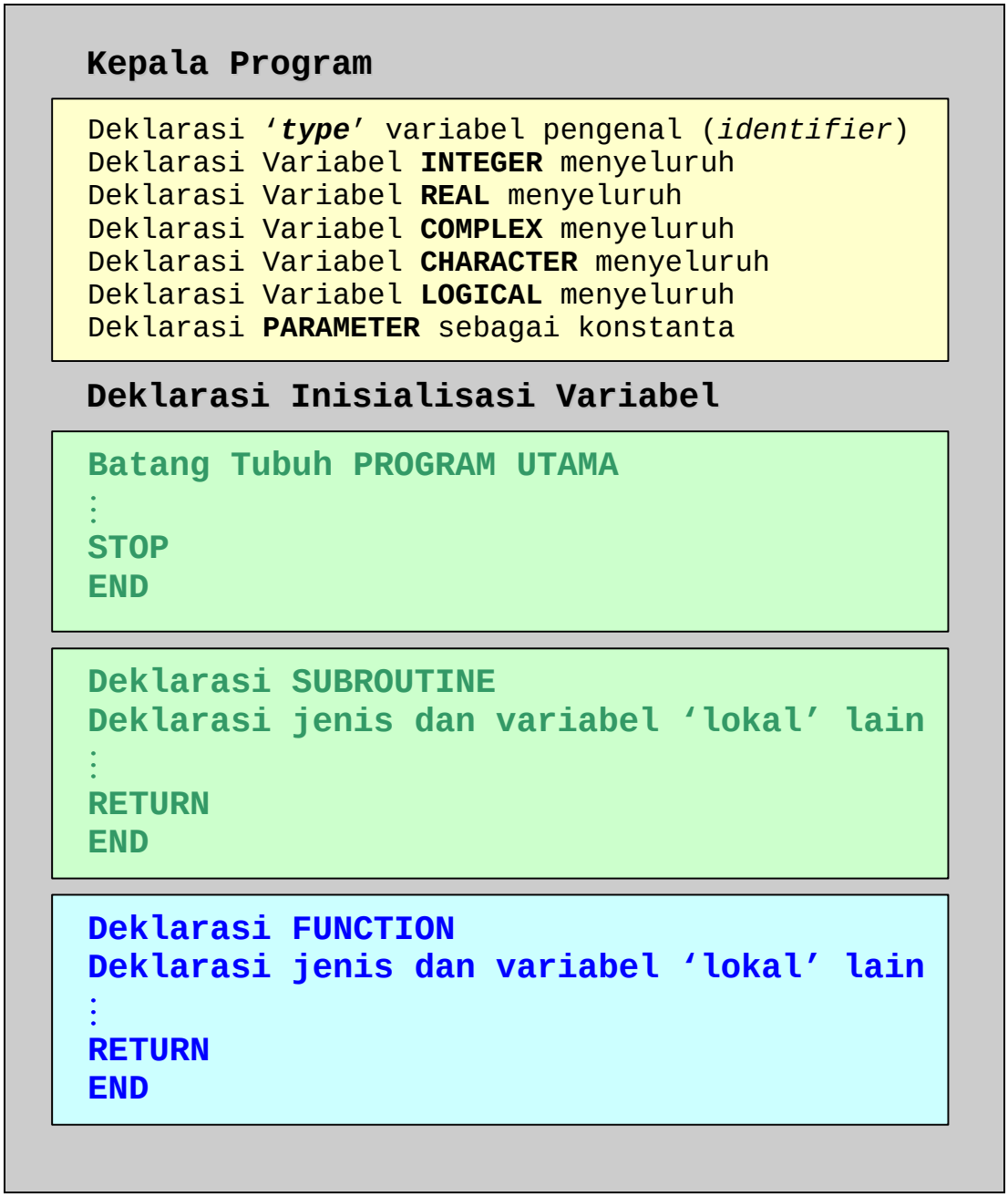
Kemudian, jembatan menuju Bahasa C ini semakin dipermudah dan jelas dengan dibuatnya ‘*compiler*’ Bahasa FORTRAN yang berbasis pada Bahasa C. Demikian pula untuk komputer-komputer kecil skala pribadi dan perkantoran (PC, *personal computer*), kecenderungan tersebut juga semakin jelas selaras dengan semakin pesatnya perkembangan platform OS yang berbasis pada UNIX, yaitu LINUX (yang dikembangkan oleh Linus Tronsvald di Eropa Utara pada akhir tahun 1980-an sampai sekarang) dan BSD (yang dikembangkan di UCSD, Davis, California).

Dewasa ini, sistem operasi LINUX telah marak, bahkan semakin menunjukkan kinerja yang memuaskan dengan dikembangkannya ‘kernel’ (inti prosedur) sistem operasi LINUX generasi 3 (kernel versi 3). Beberapa distributor LINUX terkemuka, seperti **SuSe**, **RedHat**, dan **Mandrake**, juga selalu menyertakan program FORTRAN (versi ANSI 77) yang menyatu dengan compiler Bahasa C dan C++ dalam paket-paket mereka. Lebih jauh lagi, pada saat ini sedang dikembangkan Bahasa FORTRAN 90/95 (versi ANSI 90 dan 95) untuk keperluan pemrograman dan atau komputasi numerik di dunia pendidikan tinggi.

B. Struktur dan Anatomi Bahasa FORTRAN

Sebagaimana lazimnya suatu bahasa pemrograman yang selalu memiliki struktur, anatomi dan sintaks yang khas dalam pemahamannya, demikian pula yang berhubungan dengan Bahasa

FORTRAN. Secara sederhana, struktur dan anatomi dari Bahasa FORTRAN ini dapat digambarkan seperti dalam bagan berikut:



Gambar 1. Bagan sistematis struktur dan anatomi Bahasa FORTRAN

Secara klasik, ada beberapa hal yang unik dan khas dalam pembuatan program dalam Bahasa FORTRAN, untuk itu beberapa hal yang harus diketahui dalam tata-cara penulisan program FORTRAN (terutama ANSI-77) adalah sebagai berikut:

1. Jumlah kolom (total) per-baris yang dapat ditulisi oleh kode program adalah 80 buah,
2. Jika kolom pertama diisi oleh sembarang karakter (terutama ‘C’ atau ‘*’), maka baris tersebut dialokasikan sebagai ‘*comment*’
3. Penulisan ‘nomor baris’ (*line number*) dilakukan pada kolom ke-2 sampai kolom ke-5, dengan ketentuan **rapat kanan**,
4. Penulisan baris-baris yang dapat dieksekusi (*executable lines*) dimulai pada kolom ke-7 sampai kolom ke-72,
5. Jika baris perintah di atas (butir 4) tidak mencukupi, maka jumlah baris dapat ditambah (sampai maksimum 6 baris) di bawahnya dengan cara: mengisi kolom ke-6 dengan sembarang karakter (dianjurkan diisi oleh karakter ‘*’),
6. Kolom ke-73 sampai kolom ke-80 tidak digunakan secara khusus, namun umumnya dapat diisikan oleh para pemrogram dengan kode-kode atau nomor-nomor baris yang tidak dieksekusi selama *compiling* kode program tersebut,
7. Walaupun ada beberapa compiler yang mentolerir penghapusan STOP (dalam program utama) dan RETURN (dalam subprogram), namun penulisan keduanya tetap dianjurkan,
8. Jumlah baris yang dapat dutilisi oleh kode-kode program tidak dibatasi, namun untuk OS (sistem operasi) berbasis DOS hal ini kemungkinan terbatasi oleh ‘**barier memori dasar**’ yang besarnya hanya sekitar 64 KB,
9. Untuk OS berbasis UNIX (*midi computer*), penulisan program dapat dilakukan dengan menggunakan editor ***vi***; untuk LINUX dan atau Free-BSD (PC), penulisan program dapat dilakukan dengan menggunakan editor ***vi*** dan ***emacs***,
10. Untuk OS berbasis DOS dan OS/2, penulisan program dapat dilakukan dengan menggunakan editor **edit.com**,
11. Untuk OS berbasis WINDOWS, penulisan program dapat dilakukan dengan menggunakan editor-editor **notepad**, **wordpad**, ataupun editor lainnya seperti **MS-Word**, dll.

Untuk memberikan gambaran yang lebih jelas tentang struktur dan anatomi tentang Bahasa FORTRAN, di bawah ini diberikan suatu contoh tentang pemrograman yang sejalan dengan bagan di atas:

```
C *** BARIS KOMENTAR PERTAMA                                    001
C Baris Komentar kedua *****                                    002
------
      PROGRAM Program_Satu                                      004
------
      REAL R1                                                   006
      DOUBLE PRECISION R2,R3                                    007
      REAL*8 PI,R4(2,3)                                         008
------
      INTEGER*2 I1                                              010
      INTEGER*4 I2                                              011
------
      CHARACTER C1,C2*5                                         013
      CHARACTER*12 STR                                          014
------
      LOGICAL YES,NOBODY                                        016
------
      DIMENSION R3(2,2)                                         018
      PARAMETER (PI=3.145678)                                   019
------
      OPEN(11,FILE='input.dta')                                 021
------
      R1 = 0.12345678                                           023
      R2 = 0.1234567890123456D0                                 024
      R3(1,1) = 11                                              025
      R3(1,2) = 12                                              026
      R3(2,1) = 21                                              027
      R3(2,2) = 22                                              028
------
      READ(11,*) R4                                             030
------
      I1 = 1234567890                                           032
      I2 = 1234567890                                           033
------
      C1 = 'c'                                                  035
      C2 = '12345'                                              036
      STR = '1234567890123'                                     037
------
      YES = 6 .GT. 4                                            039
      NOBODY = 0 .LT. -2                                        040
------
      WRITE(*,*) R1                                             042
      WRITE(*,*) R2                                             043
      WRITE(*,10) R2                                            044
      WRITE(*,*) R3                                             045
      WRITE(*,*) R4(1,1)                                        046
      WRITE(*,*) R4(2,1)                                        047
      WRITE(*,*) R4(1,2)                                        048
      WRITE(*,*) R4(2,2)                                        049
      WRITE(*,*) R4(1,3)                                        050
      WRITE(*,*) R4(2,3)                                        051
------
      WRITE(*,*) I1                                             052
      WRITE(*,*) I2                                             053
------
      WRITE(*,*) C1                                             055
      WRITE(*,*) c2                                             056
      WRITE(*,*) Str                                            057
------
      WRITE(*,*) Yes                                            059
      WRITE(*,*) NoBody                                         060
------
      WRITE(*,*) 'PI = ',PI                                     062
------
   10 FORMAT(5HR2 = ,F20.19)                                    064
------
      CLOSE(11)                                                 065
      STOP                                                      066
      END
```

Gambar 2. Struktur dan anatomi program dalam Bahasa FORTRAN

Perhatikan dengan seksama contoh program pada **Gb. 2** di atas, dan hasil eksekusinya adalah sebagai berikut:

```
  0.123456784
  0.123456789
R2 = .12345678901234559 41
  11.  21.  12.  22.
  1.
  2.
  3.
  4.
  5.
  6.
 1234567890
 1234567890
 C
 12345
 123456789012
 T
 F
 PI =   3.14567804
```

Gambar 3. Hasil keluaran di layar dari program FORTRAN di gambar 2.

Beberapa hal yang dapat dirinci sesuai dengan program FORTRAN dan hasil keluarannya seperti di atas adalah sebagai berikut:

(1). Penulisan komentar dapat dilakukan seperti pada baris-baris pertama ataupun kedua, asalkan kolom pertama diisi suatu karakter (dalam hal ini diisi karakter 'C').

(2). Kepala program diberi nama '**Program_Satu**' yang dituliskan pada baris ke-4. Nama kepala program tidak boleh terpisah (oleh spasi), dan harus disambung dengan karakter alfanumerik (a-z, A-Z, 0-9, dan _ atau *underscore*).

(3). Variabel R1 berjenis REAL (**4 byte**, presisi tunggal: nilai signifikan hanya 8 angka di belakang koma), perhatikanlah baris ke-23 pada **Gb**. **2** dan hasilnya pada baris pertama pada **Gb**. **3**, dimana angka ke-9 di belakang koma (= 4) sudah tidak dapat dipercaya lagi ketelitiannya.

(4). Sedangkan variabel-variabel R2, R3, R4, dan PI berjenis DOUBLE PRECISON yang identik dengan REAL*8 (8 byte, presisi ganda: nilai signifikan sampai 16 angka di belakang koma); perhatikan hasil keluaran R2 pada baris ke-2 dan ke-3 di Gb. 3, sampai angka ke-16 di belakang koma dapat dipercaya kebenarannya. Ingat pula, bahwa penambahan karakter '**D**' (bukan 'E') yang diikuti karakter '0', pada baris ke-24 pada Gb. 2 (pemasukan nilai R2) merupakan 'pernyataan konfirmasi' bahwa angka yang dimasukkan adalah berjenis **DOUBLE PRECISION**.

(5). Variabel-variabel R3 dan R4 merupakan MATRIKS (berorde 2 atau 2 dimensi). Penulisan atau pemesanan alokasi memori R3 dilakukan seperti pada baris ke-7 (untuk jenis DOUBLE PRECISION) dan diikuti dengan baris ke-18 pada Gb. 2 (untuk dimensi matriks order 2, yang dideklarasikan sebagai DIMENSION). Sedangkan pemesanan alokasi memori untuk R4 menjadi jauh lebih sederhana dan ringkas, karena cukup hanya 1 kali dilakukan yaitu pada baris ke-8.

(6). Coba perhatikan tentang pemesanan alokasi R3(2,2) dan R4(2,3), maka urutan penyimpanannya didasarkan atas 'perubahan baris dahulu' (= *i*), baru 'kemudian perubahan kolom' (= *j*), seperti diperlihatkan pada baris ke-25 sampai ke-28 pada Gb. 2 (pemasukan nilai-nilai R3) dan baris ke-4 pada Gb. 3 (pencetakan nilai R3), dan baris ke-30 pada Gb. 2 (pemasukan nilai-nilai R4) dan baris ke-5 sampai ke-10 (pencetakan nilai R4).

(7). Variabel I1 didefinisikan sebagai variabel INTEGER*2 (integer dengan panjang memori penyimpanan sebesar 2 bytes, yaitu: $-2^{15} < I1 < +2^{15} - 1$, atau $32768 < I1 < 32767$), sedangkan I2 merupakan variabel INTEGER*4 (4 bytes, biasa disingkat sebagai INTEGER saja, yaitu: $-2^{31} < I2 < +2^{31} - 1$, atau $-2147483648 < I2 < +2147483647$). Perlu dicatat bahwa: I1 tidak memberikan 'peringatan' pada saat terjadi '*overflow*', sedangkan I2 memberikan peringatan pada saat overflow.

(8). Variabel-variabel C1, C2 dan STR merupakan variabel-variabel yang dapat dikategorikan sebagai CHARACTER, hanya saja panjang masing-masingnya berbeda: C1 hanya terdiri dari 1 buah karakter saja, C2 mampu menyimpan sejumlah karakter sampai 5 buah, sedangkan STR mempunyai panjang karakter sampai 12 buah; oleh karena ini C2 dan STR dapat juga disebut sebagai variabel STRING. Perlu diperhatikan, bahwa pendefinisian panjang variabel ini dapat dituliskan pada deklarasi **CHARACTER*n** (seperti pada C2) atau pada nama variabel itu sendiri (seperti **STR*12**, yang secara umum dapat ditulis sebagai: **VAR*n**). Komputer akan memotong jumlah karakter yang terlalu panjang, dimulai dari yang paling kanan, seperti yang terlihat pada baris ke-37 pada Gb. 2 (pemasukan nilai STR) yang dikeluarkan hasilnya pada baris ke-15, pada Gb. 3.

(9). Variabel-variabel YES dan NOBODY keduanya merupakan variabel-variabel LOGICAL, yang identik dengan variabel BOOLEAN dalam Turbo PASCAL (karena hanya berharga BENAR atau SALAH).) Dalam Bahasa FORTRAN, harga variabel-variabel tersebut adalah: '**T**' (kependekan dari TRUE) atau '**F**' (identik dengan FALSE).

Perbandingan operator-operator 'nalar' (LOGICAL) maupun 'hubungan kesetaraan' (RELATIONAL) dalam program Turbo PASCAL dan FORTRAN seperti disajikan pada tabel-tabel di bawah ini:

Tabel 1. Penulisan Operator Logical (FORTRAN) dan Booloan (PASCAL).

Nama Operator	FORTRAN	PASCAL
Lebih kecil	.LT.	<
Lebih kecil atau sama dengan	.LE.	<= atau =<
Sama dengan	.EQ.	=
Tidak sama dengan	.NE.	<> atau ><
Lebih besar	.GT.	>
Lebih besar atau sama dengan	.GE.	>= atau =>

Tabel 2. Penulisan Operator Relasional dalam FORTRAN dan PASCAL.

Nama Operator	FORTRAN	PASCAL
Pengingkaran	.NOT.	not
Atau (salah satu yang benar)	.OR.	or
Dan (semuanya harus benar)	.AND.	and

(10). Pemasukan atau pembacaan satu atau beberapa data (***input data***) dilakukan menggunakan perintah:

`READ(*,*)`	pembacaan variabel kosong (sembarang), progam berhenti sampai ditekan <ENTER>
`READ(*,*)` v_1	pembacaan variabel v_1 pada *device* KEYBOARD (tanda '*' berrti menggunakan input device standar, yaitu KEYBOARD) dengan FORMAT BEBAS (yaitu tanda '*' yang kedua)
`READ(11,*)` v_1, v_2	pembacaan variabel-variabel v_1 dan v_2 pada *device* **nomor 11** (sebagai file) dengan FORMAT BEBAS
`READ(22,10)` v_3	pembacaan variabel v_3 pada *device* **nomor 22** (sebagai file) dengan FORMAT **nomor 10**
`READ(*,'(A)')` S	pembacaan variabel **S** (dalam hal ini sebagai STRING) dengan FORMAT(A), sebagai format untuk pembacaan variabel berjenis CHARACTER (A = *alphanumeric*)

Devices dengan nomor 11 atau nomor 22 seperti dituliskan pada tabel di atas, keduanya merupakan FILE yang harus dibuka terlebih dahulu, seperti di bawah ini:

```
OPEN(11,FILE='input.dta')
```

atau

```
OPEN(22,FILE='INPUT.DAT')
```

yang harus ditutup dengan perintah seperti di bawah:

```
CLOSE(11)
```

atau

```
CLOSE(22)
```

Penulisan FORMAT dengan nomor 10 seperti dituliskan pada tabel di atas, dapat berbentuk seperti di bawah:

`10 FORMAT(A)` ⇨ bila v_3 merupakan variabel CHARACTER

atau

`10 FORMAT(I3)` ⇨ bila v_3 merupakan variabel INTEGER

atau

`10 FORMAT(F12.7)` ⇨ bila v_3 merupakan variabel REAL

(11). Keluaran atau penayangan satu atau beberapa data (***output data***) dilakukan menggunakan perintah:

`WRITE(*,*)`	penulisan variabel kosong (sembarang) di monitor, yang berarti penulisan 1 baris kosong (loncat 1 spasi)
`WRITE(*,*)` v_1	penulisan variabel v_1 pada *device* layar MONITOR (tanda '*' yang di sebelah kiri menandai penggunaan output device standar, yaitu MONITOR) dengan FORMAT BEBAS (yang ditandai dengan '*' yang ke dua).
`WRITE(33,*)` v_1, v_2	penulisan variabel-variabel v_1 dan v_2 pada *device* **nomor 33** (sebagai file) dengan FORMAT BEBAS
`WRITE(33,10)` v_3	penulisan variabel v_3 pada *device* **nomor 33** (sebagai file) dengan FORMAT **nomor 10**
`WRITE(*,'(A,$)') S`	penulisan variabel **S** (dalam hal ini sebagai STRING) dengan FORMAT(A,$), sebagai format untuk penulisan variabel berjenis CHARACTER (A = *alphanumeric*) tanpa turun 1 spasi ke bawah.

Penulisan device dengan nomor 33 seperti dituliskan pada tabel di atas, sebenarnya berupa seberkas FILE yang harus dibuka terlebih dahulu, dengan perintah seperti di bawah ini:

```
OPEN(33,FILE='OUTPUT.DAT')
```

yang harus ditutup dengan perintah seperti di bawah:

```
CLOSE(33)
```

Selanjutnya, untuk memberikan gambaran yang lebih jelas tentang struktur dan anatomi subprogram-subprogram FUNCTION dan SUBROUTINE dalam Bahasa FORTRAN, di bawah ini diberikan suatu contoh:

```
      REAL*8 FVAL,FACTORIAL                              001
      INTEGER I,n                                        002
     -----
      WRITE(*,'(4Hn = ,$)')                              004
      READ(*,*) n                                        005
     -----
      FVAL = FACTORIAL(n)                                006
      WRITE(*,10) n,FVAL                                 007
     -----
      CALL SFACTORIAL(n,FVAL)                            009
      WRITE(*,10) n,FVAL                                 010
     -----
   10 FORMAT(I2,4H! = ,G18.12)                           012
     -----
      STOP                                               014
      END                                                015
     -----
     -----
      FUNCTION FACTORIAL(n)                              018
      REAL*8 FACTORIAL,FV                                019
      INTEGER n,j                                        020
     -----
      j = 1                                              022
      FV = 1.0D0                                         023
      DO WHILE (j .LE. n)                                024
         FV = FV*j                                       025
         j = j + 1                                       026
      ENDDO                                              027
     -----
      FACTORIAL = FV                                     029
     -----
      RETURN                                             031
      END                                                032
     -----
     -----
      SUBROUTINE SFACTORIAL(n,FACTORIAL)                 035
      REAL*8 FACTORIAL                                   036
      INTEGER n,j                                        037
     -----
      j = 1                                              039
      FACTORIAL = 1.0D0                                  040
      DO WHILE (j .LE. n)                                041
         FACTORIAL = FACTORIAL*j                         042
         j = j + 1                                       043
      ENDDO                                              044
     -----
      RETURN                                             046
      END                                                047
                                                         048
```

Gambar 4. Struktur dan anatomi subprogram FUNCTION dan SUBROUTINE

Perhatikan dengan seksama contoh program pada **Gb. 4** di atas, dengan hasil eksekusinya adalah sebagai berikut:

```
n = 55
55! = 0.126964033537E+74
55! = 0.126964033537E+74
```

Gambar 5. Hasil keluaran di layar dari program FORTRAN di gambar 4.

Beberapa hal yang dapat dicatat dari deklarasi subprogram-subprogram FUNCTION dan SUBROUTINE dan hasil-hasil keluarannya seperti di atas adalah sebagai berikut:

(1). Subprogram FUNCTION memiliki jenis variabel tersendiri yang harus dideklarasikan pada PROGRAM UTAMA, dalam hal ini FUNCTION yang diberi nama FACTORIAL memiliki jenis variabel REAL*8 harus didefinisikan lagi sebagai sebuah variabel dengan nama dan jenis yang sama agar supaya dapat digunakan di dalam PROGRAM UTAMA (hal ini disebut *passing by value*).

(2). Subprogram SUBROUTINE memberikan hasil proses perhitungannya melalui ARGUMEN-ARGUMENnya, dalam hal ini 'jenis' dari argumen tersebut yang harus dideklarasikan sama pada PROGRAM UTAMA. Bila diperhatikan dengan seksama, **variabel lokal** 'FACTORIAL' dalam argumen SUBROUTINE yang diberi nama SFACTORIAL memiliki jenis REAL*. Di sisi program utama, jenis REAL*8 juga harus dimiliki oleh **variabel global** yang diberi nama FVAL, sedemikian rupa sehingga hasil/keluaran SFACTORIAL dapat disalurkan melalui variabel FVAL (hal ini disebut *passing by parameter*).

(3). Penggunaan perintah DO WHILE (… *LE* …) … ENDDO merupakan salah satu perintah berulang (*looping*), yang kriteria penghentiannya menggunakan *logical expression* (*LE*), berbeda dengan DO ENDDO *loop* yang dihentikan atau dijalankan oleh **variabel penghitungnya** (yang harus berjenis INTEGER).

C. Beberapa Kendala Dalam Sistem Komputasi Numerik

Dalam komputasi secara numerik, terutama yang melibatkan bahasa-bahasa pemrograman (BASIC, Pascal, FORTRAN, C, ADA, Modula 2, dll.), selalu dijumpai beberapa kendala sistematis yang berhubungan dengan sistem kerja "prosesor" dan atau "koprosesor" dari komputer yang digunakan.

Kendala-kendala yang dijumpai umumnya berupa **sesatan** (*error*), **pembulatan** (*round-off*) dan **stabilitas** (*stability*). Di samping itu, 'problem-problem íntrinsik' yang dimiliki oleh setiap *compiler* ataupun *interpreter* bahasa pemrograman juga turut mempengaruhi kendala-kendala tersebut. Adanya kendala tersebut seringkali dapat menghambat tercapainya ***konvergensi*** solusi pendekatannya, atau bahkan sama sekali mengakibatkan arah sebaliknya, yaitu terjadinya ***divergensi***.

Secara matematis, semua problem seharusnya dapat diselesaikan, betapapun sulitnya. Pada dasarnya, solusi problem matematis tersebut dapat digolongkan dalam 2 bagian besar berikut:

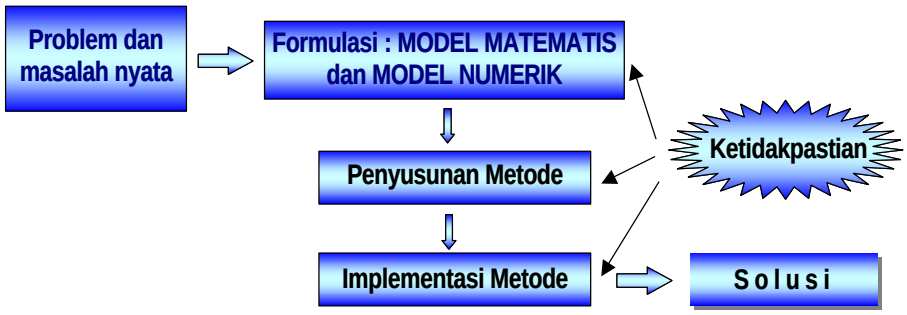
- ⇨ **Solusi EKSAK** (***exact solution***): hasil penyelesaian suatu problem matematis yang identik dengan hasil penyelesaian yang diperoleh melalui **metode analitis**.
- ⇨ **Solusi PENDEKATAN** (***approximative solution***): hasil penyelesaian suatu problem matematis dengan metode numerik yang umumnya merupakan pendekatan pada solusi eksak, karena adanya ketidakpastian dan sesatan (***uncertainty*** *and* ***errors***) dalam **proses penyelesaian problem**.

Secara umum, di dalam dunia teknik dan enjiniring lebih dipilih teknik-teknik solusi yang praktis dan menghemat waktu. Dalam hal ini, solusi pendekatan seringkali digunakan karena dianggap relatif praktis dan dapat menghemat waktu.

D. Sekuensial Proses Penyelesaian Problem

Secara sistematis, proses penyelesaian atau perolehan solusi dapat dilakukan berdasarkan urutan-urutan atau sekuens kerja berikut:

1. **Formulasi yang tepat** dari suatu model matematis dan atau pada model numerik yang sepadan, yang berarti bahwa formulasi tersebut mempunya jawab (konvergen).
2. **Penyusunan suatu metode dan atau analisis** untuk penyelesaian problem numerik, yang berarti metode yang dipilih memiliki keunggulan kecepatan, efisiensi dan efektivitas.
3. **Implementasi metode yang dipilih** dalam proses komputasi solusi/jawaban yang dipilih, yang juga bergantung pada bahasa pemrograman yang dipakai.



Gambar 6. Sekuens dari proses penyelesaian problem matematis

Di samping itu, dikenal pula 2 hal mendasar yang harus dicari simpul-simpul penghubungnya sedemikian rupa sehingga keduanya memberikan pengertian matematis dan fisik yang logis:

- ⇨ **Problem nyata**: fenomena atau proses-proses kehidupan alamiah yang dijumpai sehari-hari (gravitasi, banjir, populasi, gerakan angin, dll.),
- ⇨ **Logika Matematika:** digunakan untuk pembentukan model karena mempunyai **bahasa** dan **kerangka-kerja** yang baku.

E. Model Matematis dan Solusi Numeris

Dalam problem-problem teknik, rekayasa ataupun perancangan, pada umumnya dapat diselesaikan atau dicari solusinya secara numerik, karena 'model matematik' yang dimiliki dapat diubah menjadi 'model numeris':

- ⇨ **Model Numerik**: model yang secara numerik harus dapat diselesaikan menggunakan sejumlah tertentu tahapan-tahapan pendekatan atau perhitungan,

- ⇨ **Pendekatan Numerik**: suatu usaha untuk memudahkan pemahaman persepsi **Model Matematika** (yang bersifat analitis), dengan cara mengalihkannya menjadi **Model Numerik** (Model Matematika lainnya yang dapat diselesaikan secara numeris).

F. Solusi Numeris dan Sesatan

Seperti telah dijelaskan di atas, setiap solusi-solusi numeris yang diaplikasikan pada komputer selalu berkendala, namun demikian pada umumnya masih dapat ditoleris berdasarkan analisis galat (*error*) yang dilakukan:

1. **Sesatan Pembabatan atau pemotongan** (*truncation error*): sesatan atau kesalahan yang terjadi karena adanya pemotongan dan atau penyederhanaan proses perhitungan yang berlangsung terus-menerus dengan jumlah sangat besar (mendekati bilangan tak berhingga) ⇨ lebih bersifat Intuitif ***(mathematically unsolvable problem !***),

2. **Sesatan Pembulatan** (*round-off error*): sesatan atau kesalahan yang terjadi karena adanya pembulatan atau penyederhanaan penyimpanan bilangan yang dilakukan dalam "memori" komputer (selama penggunaan variabel-variabel dalam bahasa

pemrograman) ⇨ lebih bersifat Intrinsik (kesalahan yang tak terhindarkan karena sifat PERANGKAT KERAS !).

G. Konsep Konvergensi

Konvergensi seringkali digunakan dalam solusi-solusi numeris, sebagai parameter (alat estimasi) untuk memperkirakan bilamana problem yang dihadapi memiliki solusi atau jawab yang "mendekati solusi eksak", "dapat diterima dengan prosentase galat tertentu", atau bahkan "tidak memiliki solusi". Bila suatu problem menemui atau cenderung pada suatu "domain jawab", maka problem tersebut dapat dikatakan '***konvergen***', sedangkan bila sebaliknya, maka problem tersebut disebut '***divergen***'.

Pengertian-pengertin lain yang berhubungan dengan konvergensi ini adalah:

(a). **Order Konvergensi** (*order of convergence*): laju atau kecepatan perubahan solusi pendekatan menuju solusi eksak (dalam hal ini: sesatan pembulatan dan pemotongan juga menuju nilai sangat kecil atau menjadi nol) sebagai fungsi dari parameter-parameter metode yang dipilih. Beberapa notasi ilmiah yang sering dipakai untuk menyatakan order konvergensi adalah sebagai berikut:

1. Metode menuju konvergen setara $1/N$
2. Metode menuju konvergen setara $1/k^{3,5}$
3. Metode menuju konvergen setara h^2
4. Metode menuju konvergen secara eksponensial
5. Sesatan pemotongan berorder $1/N^5$
6. Order sesatan sebesar h^4
7. Laju konvergensi setara $(\log N)/N$

(b). **Notasi Ilmiah** (*scientific notation*): representasi angka atau penulisan bilangan dalam memori komputer berdasarkan

kaidah baku perangkat keras, misalnya: laju konvergensi "berorder $1/N^2$" berarti juga "laju perubahan ke arah jawaban eksak setara dengan $1/N^2$", yang umumnya dapat ditulis sebagai :

$$= O(1/N^2)$$

Simbol O-besar didefinisikan sebagai suatu simbol atau pengertian dengan defini sebagai berikut: Suatu fungsi $f(x)$ dikatakan sebagai $O(g(x))$ pada saat x menuju L bila

$$\lim_{x \to L} \left| \frac{f(x)}{g(x)} \right| < \infty$$

Beberapa sifat dari order dan notasi konvergensi seperti di atas seringkali mengalami perubahan menuju suatu harga tertentu berdasarkan nilai kelipatan terkecilnya:

1. $5/N^2$, $10/N^2 + 1/N^3$ dan $-6{,}2/N^2 + e^{-N}/N$ semuanya adalah $O(1/N^2)$ pada saat N menuju $L = \infty$
2. $4h$, $3h + h^2/\log h$ dan $-h + h^2 - h^3$ semuanya adalah $O(h)$ pada saat h menuju $L = 0$.

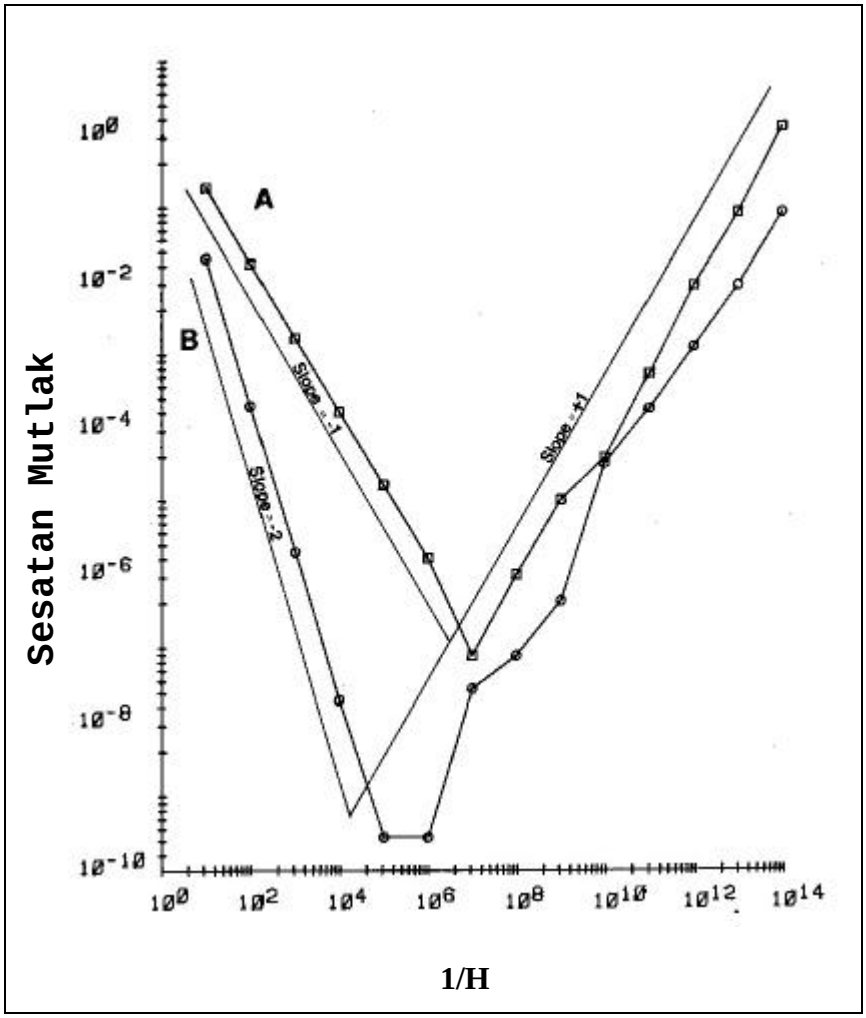
H. Contoh Soal

Coba buat skema algoritmanya (jika mungkin kode programnya), analisis pula order konvergensinya untuk persamaan-persamaan berkut:

1. Estimasi atau solusi pendekatan untuk turunan dari persamaan : $f(x) = \sin x^2$ pada $x = 5$, dengan metode:

A. $\dfrac{f(x+h)-f(x)}{h}$

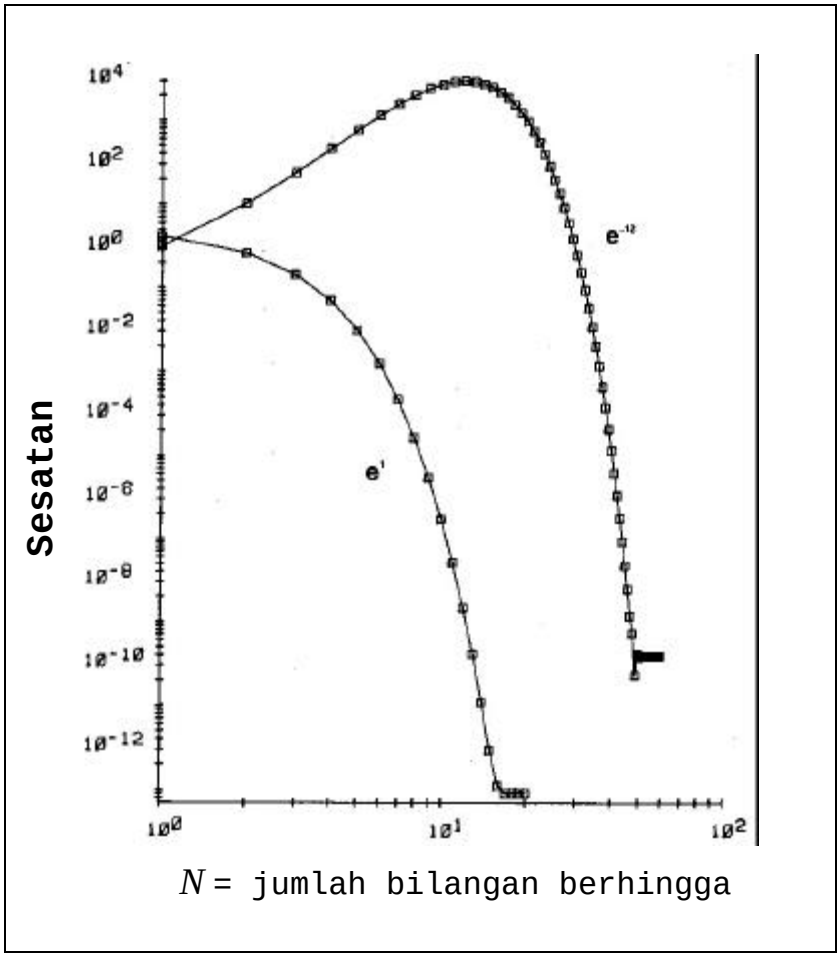
B. $\dfrac{f(x+h)-f(x-h)}{2h}$



Gambar 7. Kurva pengaluran sesatan log-log dari problem diferensiasi sederhana untuk persamaan $f(x)=\sin x^2$ pada $x=5$.

2. Analisislah propagasi harga-harga dari deret Taylor untuk e (bilangan natural) dan e^{-12}, yaitu:

$$e^x = \sum_{N=0}^{\infty} \frac{x^N}{N!}, \quad N = \text{jumlah bilangan berhingga}$$

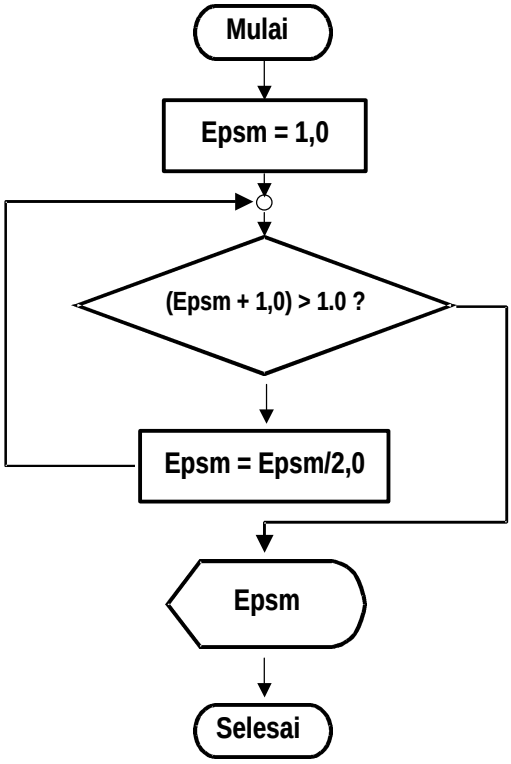


Gambar 8. Kurva pengaluran log-log sesatan dari deret Taylor untuk persamaan $f(x) = e^{x}$.

I. Pengaruh Ketelitian Mesin pada Konvergensi

Berikut ini diberikan contoh-contoh program, dalam bahasa Turbo Pascal dan FORTRAN.

Program pertama merupakan program untuk mengestimasi galat sistematis yang dimiliki suatu komputer dan programnya (disebut epsilon mesin). Diagram aliran (organigram) dari proses penghitungan ‘epsilon mesin’ tersebut adalah sebagai berikut:



Gambar 9. Diagram alir proses penghitungan epsilon mesin (**epsm**).

Listing program (*source code*) dari proses penghitungan di atas, dalam bahsa Turbo Pascal, adalah sebagai berikut:

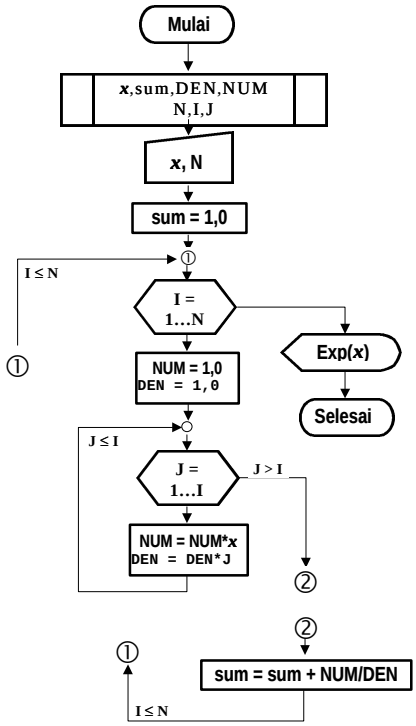
```
Var
   Epsm : Real;  {atau Extended}
Begin
   Epsm := 0.0;
   While (Epsm + 1.0) > 1.0 do
      Epsm := Epsm/2.0;
   Writeln('Epsilon Mesin = ',Epsm);
   Readln;
End.
```

Sedangkan, dalam bahsa FORTRAN (77, 90/95), adalah sebagai berikut:

```
      REAL Epsm
      C     Dapat juga dipakai REAL*8

      Epsm = 1.0D0
      DO WHILE (Epsm + 1.0D0) .GT. 1.0)
         Epsm = Epsm/2.0D0
      ENDDO
      WRITE(*,*) 'Epsilon Mesin = ',Epsm
      END
```

Program kedua merupakan program untuk mengestimasi secara numeris ungkapan ***exp***(***x***). Diagram aliran (organigram) dari proses penghitungan tersebut adalah sebagai berikut:



Gambar 10. Diagram alir proses penghitungan ***eps(x)***.

Listing program (*source code*) dari proses penghitungan di atas, dalam bahsa Turbo Pascal, adalah sebagai berikut:

```
Var
   x : Real;
   sum,NUM,DEN : Double; {atau Extended}
   N,I,J : Integer;
Begin
   Write('x = '); Readln(x);
   Write('N = '); Readln(N);
   sum := 0.0;
   For I := 1 to N do
   Begin
      NUM := 1.0; DEN := 1.0;
      For J := 1 to I do
      Begin
         NUM := NUM*x; DEN := DEN*J
      End;
      sum := sum + NUM/DEN
   End;
   Writeln('Exp(x) = ',sum);
   Readln;
End.
```

Tugas:

Salin program di atas ke dalam bahasa FORTRAN !

J. Daftar Pustaka

Atkinson, **Kendal E**., "An Introduction to Numerical Analysis", John Wiley & Sons, Toronto, 1978.

Atkinson, **L.V.**, **Harley**, **P.J.**, "An Introduction to Numerical Methods with Pascal", Addison-Wesley Publishing Co., Tokyo, 1983.

Bismo, Setijo, "Kumpulan Bahan Kuliah Metode Numerik", Jurusan TGP-FTUI, 1999.

Hanna, **O.T.**, **Sandall**, **O.C.**, "Computational Methods in Chemical Engineering", Prentice-Hall International Inc., Englewood Cliffs, New Jersey, pp. 121-149, 1995.

Press, W.H., **Flannery**, **B.P.**, **Teukolsky**, **S.A.**, dan **Vetterling**, **W.T.**, "Numerical Recipes", Cambridge Univ. Press, 1986.