

BASIC

Konsep Dasar Pemrograman BASIC

- Bahasa BASIC adalah salah satu bahasa tingkat tinggi (High Level Language) yang berorientasi ke pemecahan masalah (problem solving).
- BASIC yang merupakan singkatan dari Beginner's All purpose Symbolic Instruction Code, ditemukan oleh John G. Kemeny, profesor dari Dartmouth College dan Thomas E. Kurtz pada tahun 1960.
- Jenis compiler dari versi bahasa BASIC yang ada di pasaran : BASICA, GWBASIC, MBASIC, Turbo BASIC, Quick BASIC, Power BASIC, dll.

Data

mewakili angka, karakter dan simbol-simbol lain yang berfungsi sebagai masukan untuk proses komputer. Data yang mewakili simbol-simbol bukan merupakan informasi kecuali dalam pengertian tertentu.

- Jenis-jenis data di dalam setiap bahasa pemrograman belum tentu sama, namun biasanya terbagi menjadi beberapa bagian besar, yaitu :
 - **Data Numerik**
 - **Data String**
 - **Data Logika**
- Secara umum tipe data dibagi dalam 2 kelompok besar, yaitu :
 - a. Tipe String
 - b. Tipe Numerik

Konstanta

suatu data sifatnya tetap, dan digunakan dalam pemrograman diistilahkan dengan. Ada dua tipe konstanta yaitu :

a. Konstanta string/alphanumeric

Contoh : "Hello", "PENS", "Belajar Bahasa BASIC"

b. Konstanta numerik

- Konstanta bulat (integer)
 - **Desimal** : terdiri atas angka 0 – 9, contoh : 200, 7, 45
 - **Hexadesimal** : terdiri atas angka 0 – F, contoh : &HF00, &HFA3, &H2E
 - **Oktal** : terdiri atas angka 0 – 7, contoh : &O107,&O112
 - **Biner** : terdiri atas angka 0 – 1, contoh : &B11001100,&B11110011
- Konstanta titik tetap (real)

Contoh : 10.13, 32.123, 0.42221
- Konstanta titik mengambang (eksponensial)

Yaitu konstanta yang dituliskan dengan scientific notation.
Dengan bentuk umum : bulat.pecahan{E|D} {[+] | [-]+} pangkat
Contoh : 2.23518E+2

Variabel

Variabel (pengubah) adalah suatu lambang dari sebuah daerah di memori utama komputer yang dapat berisi suatu nilai. Variabel merupakan nama yang mewakili nilai data dimana nilai tersebut dapat berubah pada saat program dieksekusi.

Secara umum syarat-syarat penulisan nama variabel, adalah :



- a. Nama variabel jangan terlalu panjang, meskipun harus dengan jelas menunjukkan fungsi nilai data yang diwakilinya. (sebab setiap bahasa pemrograman mempunyai batas maksimal panjang nama variabel).
- b. Nama variabel tidak menggunakan tanda-tanda khusus seperti tanda baca dan spasi; meskipun dalam bahasa pemrograman tertentu dapat digunakan suatu pemisah dalam penulisan nama variabel. Dalam BASIC adalah titik (.). Misal : Nama.Siswa\$

Jenis Variabel

Di dalam bahasa BASIC suatu variabel dibedakan atas :

Variabel numerik adalah variabel yang mengandung nilai numerik atau angka

Variabel string adalah variabel yang berisi nilai huruf /alpha-numerik

Penggolongan variabel dalam bahasa BASIC dijelaskan pada tabel berikut ini :

Jenis variabel	Simbol	Range nilai	Memory yang dibutuhkan
String	\$		3 byte
Integer	%	-32768 s/d 32767	2 byte
Long Integer	&	-2.147.483.648 s/d -2.147.483.647	4 byte
Single precision	!	$\pm 8.43 \times 10^{-37}$ s/d $\pm 3.37 \times 10^{38}$	4 byte
Double precision	#	$\pm 4.19 \times 10^{-307}$ s/d $\pm 1.67 \times 10^{308}$	8 byte

Variabel numerik

A. Single Precision

Variabel presisi tunggal (single precision) merupakan default dari variabel yang digunakan dalam bahasa BASIC. Jadi kalau membuat suatu variabel dan tidak ditambahkan karakter apapun (!,#,%,&,\$) berarti variabel tersebut bertipe presisi tunggal. Selain itu variabel ini juga biasanya ditulis dengan tambahan karakter '!' di belakangnya. Variabel ini tergolong variabel yang dapat menampung bilangan real (pecahan) dan membutuhkan memory sebesar 4 byte. Variabel jenis ini mempunyai ketepatan sampai dengan 7 digit.

Program 2.1.

'Variabel bertipe single precision

'ditulis dengan tanda !

a!=100

print a!

'Variabel single precision dapat juga

'ditulis tanpa tanda !

x= 22

y= 7

z= x/y

? z



Output :

33.33333206176758 yang dapat dipercaya (significant)

3.142857074737549 hanya 7 digit pertama

B. Double Precision

Variabel bertipe ini mempunyai ketepatan sampai dengan 15 digit. Variabel ini selalu diakhiri dengan tanda '#' dan membutuhkan memory sebesar 8 byte.

Program 2.2.

'Variabel bertipe double precision

a#=100

b#=3

c#=a# / b#

print c#

x#= 22

y#= 7

z#= x#/y#

? z#

Output :

33.333333333333334 yang dapat dipercaya (significant)

3.142857142857143 sampai dengan 15 digit pertama

C. Integer

Variabel integer adalah variabel numerik yang dapat menampung bilangan bulat (tidak mengandung pecahan) dari -32768 sampai dengan 32767. Bila terdapat nilai pecahan maka akan dibulatkan. Pembulatannya adalah jika lebih besar atau sama dengan 5 maka akan dibulatkan ke atas, sedangkan jika kurang dari 5 akan dibulatkan ke bawah. Variabel jenis ini hanya membutuhkan memory sebesar 2 byte dan penulisannya selalu menggunakan tanda '%'.
Program 2.3.

'Variabel bertipe integer

cls

a% = 22

b% = 7

c% = 22/7

?c%

d%= 5/4

?d%

e%= 10/4

?e%

Output

3 → pembulatan ke bawah

1 → hasilnya mestinya 1.25 dibulatkan ke bawah

2 → hasilnya mestinya 2.5 dibulatkan ke atas

D. Long Integer

Variabel ini juga hanya menyimpan bilangan bulat tetapi mempunyai jangkauan nilai yang jauh lebih besar daripada variabel bertipe integer. Variabel ini selalu diakhiri dengan tanda '&' dan membutuhkan memory sebesar 4 byte.

Program 2.4.

'Perbandingan variabel integer dan long integer

a= 100

b= 1000

c& = a*b

?c&

Output :

100000 (tipe long integer)



d% = a*b
?d%

Terjadi overflow / error karena variabel integer maksimal hanya mampu menampung nilai -32768 s/d 32767

Variabel string

Variabel string disebut juga dengan variabel alphanumeric. Variabel ini selalu menggunakan tanda '\$' dan membutuhkan memory sebesar 3 byte.

Program 2.5.

'Contoh variabel string

```
cls
nama$="Adi"
kelas$="11A01"
print "Namaku ";nama$;" kelas ";kelas$
a$="Teknik"
b$="Informatika"
? a$+" "+b$+"..."
```

Output

Namaku Adi kelas 11A01
Teknik Informatika...

Operator

Operator adalah simbol-simbol khusus yang digunakan untuk mengoperasikan suatu nilai data (operand).

Ada beberapa jenis operator, yaitu :

a. Operator Aritmatika

Digunakan untuk mengoperasikan data-data numerik, seperti penjumlahan, pengurangan, perkalian, pembagian, dll. Dalam proses aritmatika tersebut, pengerjaan operasi tergantung dari tingkat valensi operator-operator yang terlibat. Perpangkatan memiliki valensi tertinggi, kemudian dilanjutkan dengan perkalian, pembagian, pembagian bulat dan sisa pembagian, sedangkan penjumlahan dan pengurangan mempunyai valensi yang terendah.

Operator	Keterangan	Hirarki
^	Pangkat	1
*	Perkalian	2
/	Pembagian real	2
\	Pembagian integer	2
MOD	Modulus (sisa pembagian)	2
+	Penjumlahan	3
-	Pengurangan	3

b. Operator Relasi

Digunakan untuk mewakili sebuah nilai logika (nilai boolean), dari suatu persamaan atau nilai.

Operator-operator yang terlibat adalah :

- = : sama dengan
- > : lebih besar
- < : lebih kecil
- <> : tidak sama dengan



\geq : lebih besar atau sama dengan

\leq : kurang atau sama dengan

c. Operator Logika

Digunakan untuk mengoperasikan operand (konstanta, variabel, atau suatu ekspresi) secara logis. Operator-operator logika yang umum dalam bahasa pemrograman : AND , OR, NOT.

Ungkapan (Ekspresi)

Ungkapan dapat berupa konstanta (untai/numerik), variabel (untai/numerik) dan nilai tunggal yang diperoleh dengan mengkombinasikan operand dan operator, seperti $5+4$.

Ungkapan-ungkapan dibagi menjadi empat kategori :

a. Ungkapan numerik

$2 + 5 \cdot 3^4 + 7^5$

b. Ungkapan string

"ABCD" + "EFGH" A\$ + B\$

Satu-satunya operator yang berlaku pada ungkapan string hanyalah tanda +, yang berfungsi untuk menggabungkan dua untai.

c. Ungkapan relasi/hubungan

Tipe untai dapat juga menggunakan operator relasi seperti halnya dengan tipe numerik. Misalnya diketahui bahwa 'A' lebih kecil dari 'B'

d. Ungkapan logika

NOT (A)

$A > 5$ AND $B = 4$

A\$="Agus" OR A\$="Doni"



Perintah-perintah Dasar

Perintah-perintah Input

Perintah-perintah input digunakan untuk memasukkan data ke dalam program melalui perangkat luar seperti keyboard, mouse, file, dll. Diantaranya adalah perintah INPUT, LINE INPUT, READ-DATA dan RESTORE

INPUT

Perintah INPUT digunakan untuk menerima masukan data dari keyboard. Variasi penggunaan dan cara penulisan perintah INPUT adalah sbb. :

```
INPUT a
INPUT x,y
INPUT "Masukkan alas : ",alas
INPUT "Masukkan alas,tinggi :",alas,tinggi
INPUT "Masukkan kata : ", kata$
```

Jika pada perintah input tidak ditambahkan pesan (yang tertulis di antara dua tanda petik), pada saat program di-RUN akan keluar tanda tanya yang artinya komputer menunggu pemakai program untuk memasukkan data yang diinginkan. Untuk mengakhiri proses pemasukan data harus ditekan tombol ENTER. Perintah INPUT dapat digunakan untuk memasukkan data numerik (angka) maupun string.

LINE INPUT

Perintah LINE INPUT digunakan untuk memasukkan suatu nilai string ke variabel string. Kelebihan perintah ini dibandingkan perintah INPUT biasa adalah bahwa perintah ini dapat digunakan untuk memasukkan rangkaian string yang dipisahkan oleh tanda koma. Tetapi perintah LINE INPUT tidak dapat digunakan untuk memasukkan data numerik.

Contoh perbandingan perintah INPUT dan LINE INPUT :

```
REM Menggunakan perintah INPUT
INPUT "Masukkan nama : ",nama$
PRINT nama$
```

Output

```
Masukkan nama : satu,dua,tiga
satu output tidak sesuai dengan input
```

```
REM Menggunakan perintah LINE INPUT
LINE INPUT "Masukkan nama : ",nama$
PRINT nama$
```

Output

```
Masukkan nama : satu,dua,tiga
satu,dua,tiga
```

READ-DATA

Perintah READ dan DATA saling berhubungan. READ membaca data yang didefinisikan oleh perintah DATA. Jumlah perintah READ tidak boleh lebih banyak dari pada perintah DATA. Letak perintah READ boleh di atas perintah DATA maupun di bawahnya.

Program 2.

```
Data "Agus",75,"Budi",68
Data "Dewi",90
```



```

print nama$,nilai
Read nama$,nilai
print nama$,nilai
Read nama$,nilai
print nama$,nilai
Read nama$,nilai
print nama$,nilai
Data "Eko",65

```

Output :

```

Agus 75
Budi 68
Dewi 90
Eko 65

```

RESTORE

Nilai yang tertera pada instruksi DATA hanya dapat dibaca satu kali saja dengan instruksi READ. Agar nilai tersebut dapat dibaca kembali maka dapat digunakan instruksi RESTORE.

Program

```

Data "Agus",75
Read nama$,nilai
? nama$,nilai
Read nama$,nilai out of data
? nama$,nilai

```

Jika program di atas dijalankan maka akan terjadi kesalahan (error) yaitu jumlah data lebih sedikit dari pada jumlah perintah READ, sehingga pada saat dijalankan perintah READ yang kedua terjadi 'out of data'. Untuk mengatasi hal itu dapat digunakan perintah RESTORE

Program

```

Data "Agus",75
Read nama$,nilai
? nama$,nilai
restore
Read nama$,nilai
? nama$,nilai

```

RUN

```

Agus 75
Agus 75

```

Perintah-perintah Output

PRINT

Instruksi output merupakan instruksi yang digunakan untuk menampilkan hasil pengolahan data oleh komputer, baik melalui monitor, printer maupun media lainnya.

Program

```

? 1;2;3
? "satu";"dua";"tiga"
? 1,2,3
? 1,,2
? "satu","dua","tiga"

```



```
? "satu" "dua" "tiga"
? 10 20 30
Output :
1 2 3
satuduatiga
1 2 3
1 2
satu dua tiga
satuduatiga
10 20 30
```

PRINT TAB

Untuk mengatur jarak dari nilai yang akan dicetak di layar dapat digunakan perintah PRINT TAB. Perintah ini menggunakan nilai 1 s/d 80

Program :

```
print "1234567890123456789012345678901234567890"
print tab(1);"Belajar";tab(10);"Program";tab(25);"BASIC"
RUN
1234567890123456789012345678901234567890
Belajar Program BASIC
```

PRINT USING

Perintah PRINT USING dapat digunakan untuk mengatur banyak digit yang ingin ditampilkan di layar.

Tanda yang digunakan dalam perintah PRINT USING #	Menampilkan bilangan sebanyak dengan digit sebanyak tanda #
+#	Menampilkan tanda + jika bilangannya positif
\$#	Menampilkan tanda dollar di awal bilangan
### ^{^^}	Tampilkan output dengan notasi eksponen
*#	Menampilkan tanda * pada sisa spasi kosong di awal bilangan
###,###	Menentukan pemisah digit ribuan dengan tanda ,
###.##	Menentukan jumlah digit pecahan

Program :

```
a=100000000
print using "###,###,###,###.##";a
print using "##.###";10/3
print using "$###,###.##";12500
print using "*###,###,###,###.##";a
print using "#.###^^^";1234000
```

Output :

```
100,000,000.00
```




```
3.333
$12,500.00
*****100,000,000.00
1. 23E+06
2.
```

LPRINT

Perintah LPRINT digunakan untuk menampilkan keluaran program ke printer. Tata cara penulisan sama persis dengan perintah PRINT biasa, hanya saja keluarannya tidak ditampilkan di monitor tetapi di printer.

