

# **TIPE, OPERATOR, dan EKSPRESI**

# **TIPE, OPERATOR, dan EKSPRESI**

## **Tipe Dasar**

Tipe data terdiri dari tipe:

1. **Tipe dasar**

Tipe yang dapat langsung dipakai (disediakan oleh bahasa pemrograman).

**Contoh :** Boolean, Integer, Real, Char, String.

1. **Tipe bentukan**

Tipe yang didefinisikan sendiri oleh pemrogram. Tipe yang dibentuk dari tipe dasar atau dari tipe bentukan lain yang sudah didefinisikan.

**Contoh :** Tipe Dasar yang diberi nama Tipe Baru, Record.

## **Bilangan Logik**

Nama tipe bilangan ini adalah **Boolean**, ranah nilai boolean adalah **true** dan **false**. Operator yang digunakan **NOT, AND, OR,** dan **XOR**.

**Contoh :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A B A | **NOT** B A | **AND** B A | **OR** B A | **XOR** B | B |
| True | True | False | True | True | False |
| True | False | True | False | True | True |
| False | True | True | False | True | True |
| False | False | True | True | False | False |

## **Bilangin Bulat**

Nama tipe Integer. Ranah Nilai. Secara teoritis bilangan bulat tidak mempunyai batasan untuk nilai. Tetapi pada dunia pemograman, ranah nilai bilangan bulat mempunyai batasan. Hal tersebut dimaksudkan untuk penghematan alokasi memori. Berikut rentang nilai pada Pascal dan C.

**Pascal Rentang Nilai C**

**Byte** 0 … 255 Unsigned Char

**Shortint** -128 … 127 Signed Char

**Word** 0 … 65535 Unsigned int

**Integer** -32768 … 32767 Int, Shortint

**Longint**  2147483648 … 2147483647 Longint

Setiap bilangan bulat mempunyai keturunan, atau dengan kata lain sebuah bilangan bulat mempunyai nilai sembarangan *(predecessor)* dan nilai sesudahnya *(succesor)*.

Misalnya : Nilai 8 memiliki *predecessor* 7 dan *successor* 9.

Konstanta untuk bilangan bulat tidak boleh mengandung titik desimal. Operasi :

* 1. **Operasi Aritmatika, yaitu**

**+ (tambah)**

**- (kurang)**

**\* (kali)**

**div (bagi)** dengan hasil bilangan bulat tanpa titik desimal

**mod (sisa hasil bagi)**

Keterangan : Objek yang di operasikan disebut operand. Misalnya : a + b artinya a dan b merupakan operand sedangkan + adalah operator.

* 1. **Operasi Perbandingan.**

Hasil operasi perbandingan akan menghasilkan nilai boolean (*true* dan *false*).

**< (lebih kecil)**

**≤ (lebih kecil atau sama dengan)**

**> (lebih besar)**

**≥ (lebih besar atau sama dengan)**

**= (Sama dengan)**

**≠ (Tidak sama dengan)**

## **Bilangan Riil**

Bilangan riil adalah bilangan yang mengandung pecahan desimal, misalnya 3.65, 0.0003, 29.0, 2.60240000E-6, .24 , dll. Semua konstanta bilangan riil harus mengandung titik (desimal). Misalnya “18” dianggap sebagai bilangan bulat, tetapi “18.0” dianggap sebagai bilangan riil. Untuk bilangan riil yang ditulis dengan notasi E berarti perpangkatan sepuluh.

**Contoh :** 2.60240000E-6 artinya 2.60240000×10-6.

Nama tipe bilangan riil adalah real. Ranah nilai bilangan riil sama dengan bilangan bulat, tetapi penerapan pada bahasa pemograman berbeda. Untuk bahasa **PASCAL**, penerapannya :

**Tipe Rentang Nilai Format**

**real** 2.9×10-39 … 1.7 x 1038 6 Byte

**Single** 1.5 x 10-45 …. 3.4 x 1038 4 Byte

**Double** 5.0 x 10-324 … 1.7 x 10308 8 Byte

**Extended**  3.4 x 10-4932 … 1.1 x 104932 10 Byte

Sementara, dalam bahasa C, hanya ada 2 tipe untuk bilanganriil yaitu **float** dan **double**. **Float** adalah bilangan riil berpresisi tunggal *(single-precision)*, **Double** adalah bilangan riil berpresisi ganda *(double-precision)*. Sementara untuk *long double* mengspesifikasikan tipe bilangan riil yang berpresisi-extended.

Operasi pada bilangan riil ada 2 macam, yaitu :

1. **Operasi Aritmatika, yaitu**

**+ (tambah)**

**- (kurang)**

**\* (kali)**

**div (bagi)** dengan hasil bilangan bulat tanpa titik desimal

**mod (sisa hasil bagi)**

Keterangan : Jika dilakukan operasi campuran dengan salah satunya operandnya bertipe bilangan riil sedangkan operandnya yang lain bilangan bulat, maka otomatis operand yang bilangan bulat akan otomatis di konversi ke bilangan riil.

1. **Operasi Perbandingan.**

Hasil operasi perbandingan akan menghasilkan nilai boolean (*true* dan *false*).

**< (lebih kecil)**

**≤ (lebih kecil atau sama dengan)**

**> (lebih besar)**

**≥ (lebih besar atau sama dengan)**

**= (Sama dengan)**

**≠ (Tidak sama dengan)**

Keterangan : Pada bilangan riil tidak dikenal operasi sama dengan “=”. Karena bilangan riil tidak dapat disajikan secara tepat oleh komputer. Misalnya 1/3 tidak sama dengan 0.33333, sebab 1/3 = 0.3333333…. (dengan angka 3 yang tidak pernah berhenti).

## **Karakter**

Yang termasuk karakter adalah semua huruf-huruf abjad, semua tanda baca, angka ‘0’, ‘1’, … , ‘9’. Dan karakter-karakter khusus seperti ‘&’, ‘^’, ‘%’, ‘#’, ‘@’, dll. Karakter kosong (null) adalah karakter yang panjangnya nol dan dilambangkan ‘ ‘.

Nama tipe untuk karakter adalah **char**. Ranah nilai untuk karakter adalah semua huruf di dalam alfabet (‘a’, … ‘z’, ‘A’ … ‘Z’) angka desimal (0 … 9), tanda baca (‘.’, ‘:’, ‘!’, ‘?’, ‘,’, dll), operator aritmetik (‘+’, ‘-‘, ‘\*’, ‘/’), dan karakter-karakter khusus seperti (‘$’, ‘#’, ‘@’, ‘^’, ‘~’, dll). Daftar karakter baku yang lengkap dapat dilihat di dalam buku-buku yang memuat tabel ASCII.

Konstanta karakter harus diapit oleh tanda petik tunggal. Ket: ingatlah bahwa ‘9’ adalah karakter, tetapi 9 adalah integer. Untuk operasi yang dilakukan hanya operasi perbandingan yaitu :

**< (lebih kecil)**

**≤ (lebih kecil atau sama dengan)**

**> (lebih besar)**

**≥ (lebih besar atau sama dengan)**

**= (Sama dengan)**

**≠ (Tidak sama dengan)**

Keterangan : Tipe karakter juga mempunyai keterurutan *(successor dan predecessor)* yang ditentukan oleh cara pengkodeannya didalam komputer, misalnya pengkodean ASCII. Operasi perbandingan menghasilkan nilai boolean.

## **String**

String adalah untaian karakter dengan panjang tertentu. Nama tipe string adalah string. Ranah nilai tipe string adalah deretan karakter yang telah didefenisikan pada ranah karakter. Konstanta pada tipe string harus diapit oleh dua buah tanda petik tunggal.

**Contoh :**

‘Makassar’

‘boby’

‘Abc1234’

‘\*…\*…#008\*’

‘B’

Untuk string kosong (null) dilambangkan dengan ‘ ’.

Operasi pada tipe string didefinisikan 2 macam :

1. **Operasi Penyambungan *(concatenation)***

Operasi “+”. Operator ini bukanlah operator penjumlahan seperti pada tipe real/integer tetapi operator penyambungan.

**Contoh :**

‘Boby’ + ‘Hermez’ = ‘Boby Hermez’.

1. **Operator Perbandingan**

Sama halnya tipe karakter akan menghasilkan nilai boolean =, ≠, <, ≤, >, ≥.

## **Tipe Bentukan**

Tipe bentukan adalah tipe yang didefenisikan sendiri oleh pemogram (user defined type data). Tipe bentukan di susun oleh satu atau lebih tipe dasar. Ada 2 macam tipe bentukan :

1. Tipe dasar yang diberi dengan nama tipe baru
2. Tipe terstruktur

Tipe dasar yang diberi nama tipe baru *(user defined type data)*. Penamaan tipe baru disebabkan oleh kemungkinan untuk memudahkan interpretasi dan pembacaan program oleh pemogram. Kita dapat memberi nama baru untuk tipe dasar tersebut dengan kata kunci tipe. Untuk nilai, Ranah nilai, konstanta, dan operasi-operasi tipe tersebut tidak berubah dan sesuai dengan tipe dasar aslinya, misalnya : tipe bilangan bulat / integer.

Keterangan : Bilangan sama saya dengan tipe integer. Jadi, misalkan variabel i bertipe bilangan bulat, sama saja jika i bertipe integer.

## **Tipe Dasar yang Diberi Nama Baru**

1. Nama Fungsi :

**FUNCTION** Maksimum (input A,B : integer) 🡪 integer

*{ mengembalikan nilai terbesar antara A dan B }*

*Maksimum adalah nama fungsi*

1. Nama prosedur :

**PROCEDURE** Tukar(input/output A, B : integer)

*{ mempertukarkan nilai A dan B }*

*Tukar adalah nama prosedur*

## **Tipe Terstruktur**

Tipe terstruktur adalah tipe yang berbentuk rekaman *(record)*. Rekaman *(record)* disusun oleh satu atau lebih *field*. Tiap *field* menyimpan data dari tipe dasar tertentu atau dari tipe bentukan lain yang sudah didefinisikan sebelumnya. Nama rekaman ditentukan sendiri oleh pemogram.

**Contoh :**

Mendefenisikan titik *(type)*

Titik dalam koordinat kartesian dinyatakan sebagai (x, y).

Penulisannya :

tipe titik : *record* < x : real, y : real > ; atau

tipe titik : *record* < x, y : real >

Jika pendeklarasian P adalah variabel yang bertipe titik makan pendeklarasian tiap field pada P adalah :

P. x

P. y

## **Nilai**

Nilai adalah besaran dari tipe data yang sudah didefinisikan (tipe dasar dan tipe bentukan). Nilai dapat berupa isi yang disimpan oleh nama peubah atau nama konstanta, nilai dari hasil perhitungan, atau nilai yang dikirim oleh fungsi. Nilai dapat dimanipulasi dengan cara mengisikannya ke perubahan lain dengan tipe sama, di pakai untuk perhitungan atau dituliskan kepiranti keluaran.  
Pengisian nilai kedalam peubah dilakukan dengan 2 cara, yaitu :

* + 1. **Pengisian Nilai Secara Langsung**

Pengisian nilai secara langsung adalah memasukkan sebuah nilai ke dalam nama peubah langsung di dalam teks algoritma dengan syarat nilai yang diisikan harus bertipe sama dengan dengan tipe peubah.  
Akibat dari pengisian ini, nilai peubah akan hilang dan tergantikan dengan nilai yang baru dimasukkan.

* + 1. **Pengisian Nilai Lewat Pembacaan**

Pengisian nilai lewat pembacaan adalah pengisian nilai dari piranti masukkan. Dalam algoritma, pernyataan pembacaan adalah dengan menggunakan notasi *“read”*.

## **Pengisian Nilai Secara Langsung**

Nilai adalah besaran dari tipe data yang terdefinisi (tipe dasar atau tipe bentukan). Dapat berupa data yg disimpan di dalam peubah atau konstanta, nilai dari hasil hitung, atau nilai yg dikirim oleh fungsi. Pada ALgoritma, memanipulasi nilai di dalam peubah/variabel (yg bertipe sama).

**Pengisian secara langsung (*assignment*) :**

Mengisi sebuah nilai ke dalam peubah secara langsung (nilai yg diisikan harus bertipe sama dg tipe peubah).

**Contoh :**

M 🡨 16

P 🡨 M \* 2

M 🡨 P + 100

P 🡨 P + M

## **Pembacaan**

Nilai peubah dapat diisi secara eksternal dafri piranti masukan, misalnya dari *keyboard*, dari *file*, *mouse*, dsb. Akan membaca sesuai apa yg kita masukkan (inputkan).

**Contoh :**

**Read (M) :** Komputer membaca nilai M yg diinputkan dari luar (mis: keyboard), jika nilai yg diketik adalah 78, maka lokasi memori yg bernama M sekarang berisi 78.

## **Ekspresi**

Ekspresi  adalah transformasi nilai menjadi keluaran yang dilakukan melalui suatu perhitungan (komputasi). Ekspresi terdiri atas **operand dan operator**, contoh ekspresi : **‘a + b’**. Hasil Evaluasi dari sebuah Ekpresi adalah nilai yang sesuai dengan tipe operand yang dipakai.

## **Ekspresi Aritmetik**

Ekspresi yang baik operand-nya bertipe numerik dan hasilnya juga bertipe numerik. Ekspresi aritmatika memakai operator aritmatika. Pada ekspresi aritmetik terdapat 2 buah operator yaitu :

Operator biner, yaitu ekspresi yang operatornya membutuhkan 2 buah operand.

**Contoh** **:** a + b

Operator Uner, yaitu “-” atau operator yang punya 1 operand contoh “-2”

**Contoh :** Penulisan ekspresi dengan notasi algoritma

T = 5/9 \* (c + 32)

Z = (2\*x + y) / (5 \* w)

Y = 5((a+b) / (c\*d) + m (p + q))

## **Ekspresi Realsional**

Ekspresi relasional adalah ekspresi dengan operator <, ≤, >, ≥, =, dan ≠, not, and, or dan xor dengan menghasilkan nilai bertipe boolean (true atau false). Biasanya ekspresi Relasional disebut ekspresi boolean.

## **Ekspresi String**

Adalah sebuah perintah yang akan mengeksekusi suatu perintah apabila kondisi tertenti  terpenuhi. Operator boolean digunakan untuk menghubungkan dua buah ekspresi logika. Pada kesempatan ini operator boolean yang akan dibahas adalah **AND** dan **OR**. Operator AND digunakan untuk menghubungkan dua buah ekspresi logika dan hasilnya **TRUE** jika kedua ekspresi tersebut bernilai TRUE.

## **Menuliskan Nilai ke Piranti Keluaran**

Nilai yang disimpan oleh memori dapat ditampilkan ke piranti keluaran (misalnya monitor). Instruksi penulisan nilai dilakukan dengan notasi **write**. Dengan catatan bahwa nama1, nama2, ..., namaN dapat berupa nama peubah atau nama tetapan. Dengan instruksi penulisan ini, nilai yang disimpan di dalam memori dituliskan/dicetakkan ke piranti keluaran.

write(nama1, nama2, ..., namaN)

write(tetapan)

write(nama, tetapan, ekspresi)

write(ekspresi)