

# Nyankod Magz

Love it and Earn It

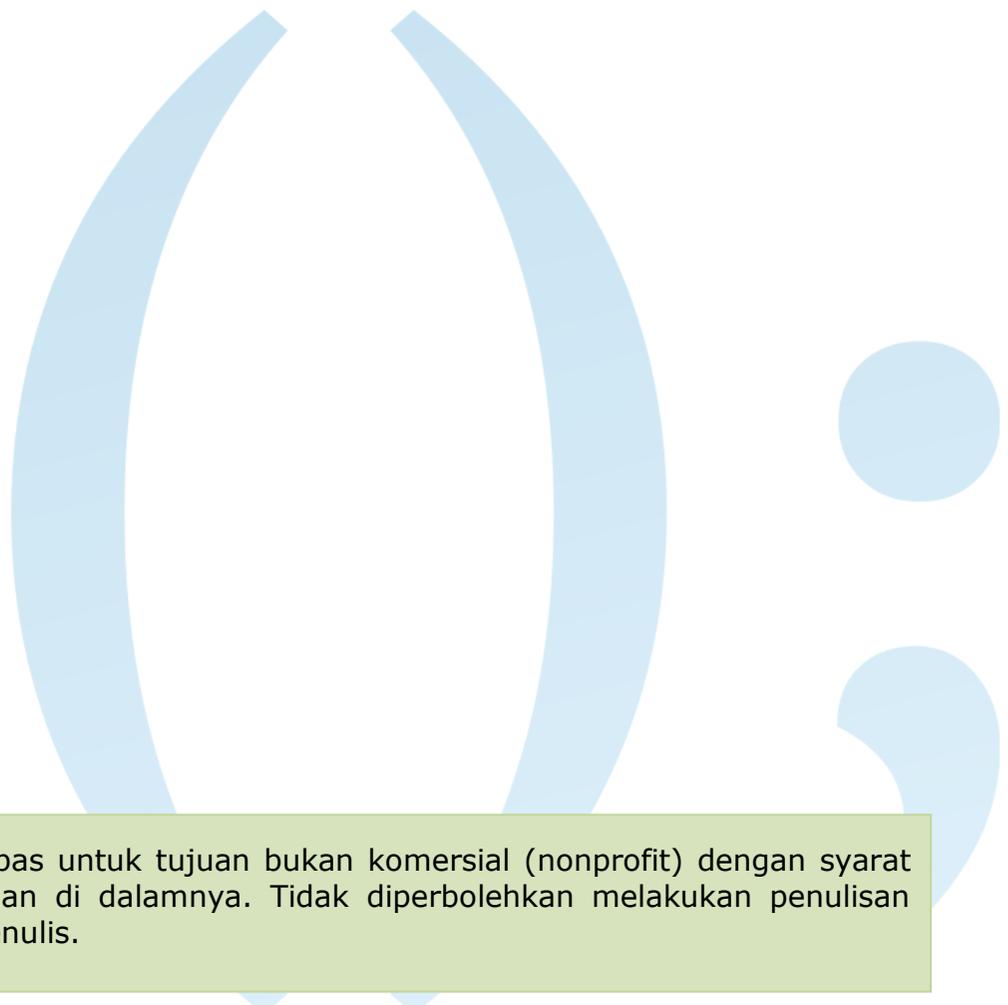
## Operator





# **Nyankod Magz Edisi 3 Operator**

Love it and Earn It



Nyankod Magz ini dapat Anda sebarkan secara bebas untuk tujuan bukan komersial (nonprofit) dengan syarat tidak mengurangi dan menambahkan isi kandungan di dalamnya. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapat izin terlebih dahulu dari penulis.

# Sapa Nyankod

Salam coder buat Nyankoders se-Indonesia...

Kembali lagi kami Nyankodist Team menghadirkan sajian untuk Anda semua, Nyankod Magz edisi ke-3. So... jangan lupa siapkan kopi hangat plus gorengan dan beberapa list musik dangdut buat menemani membaca. Nyankod Magz kali ini kami akab berbicara tentang berbagai operator dan operasi yang ada di masing-masing bahasa pemrograman. Selamat menikmati ya jou...

Oh ya, seperti kata pepatah ada udang di balik tong sampah, artinya?? Ah bodo amat... Yang penting seneng banget deh kita udah bisa nerbitin sampe edisi ke-3. Semoga karya ini bisa bermanfaat khususnya buat para temen-temen yang lagi belajar pemrograman.

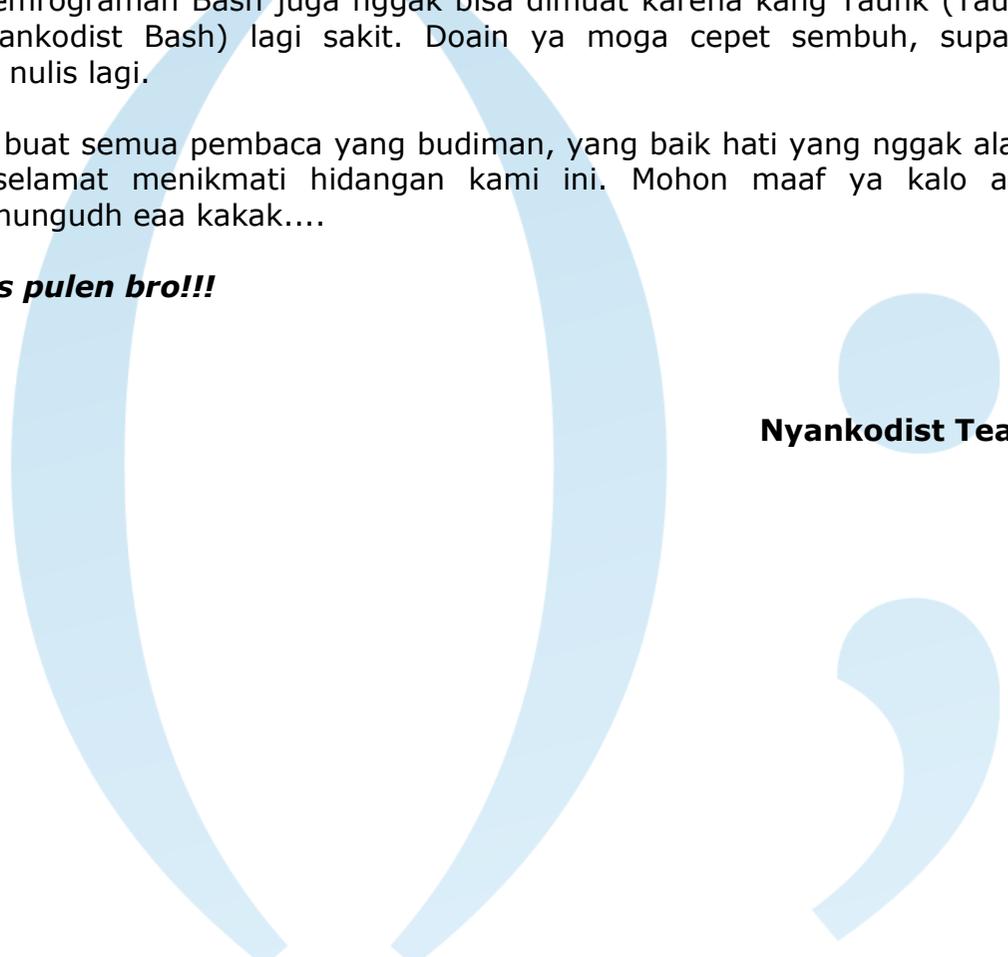
Sekalian di sini kami mau ngucapin terima kasih atas apresiasi dari semua temen-temen yang udah ngedukung project Nyankod ini. Oh ya, kemarin ada dari pembaca yang bertanya kepada kami mengenai tambahan bahasa pemrograman. Ada pembaca yang minta ditambahkan pembahasan mengenai VB.net, ada juga yang minta Assembly. Pengen rasanya kami juga senantiasa menambahkan pembahasan biar lebih rame, lengkap dan lebih baik lagi. Namun apa daya, untuk saat ini, sumber daya penulis masih belum memadai untuk tambahan bahasa pemrograman. Mudah-mudahan kedepannya kami bisa mendapatkan kontributor tambahan buat gabung jadi tim nyankodist. So request pembaca mengenai beberapa bahasa pemrograman lainnya bisa terpenuhi.

Jalan kadang emang nggak selalu lurus ya jou, kadang berliku-liku, malahan kadang naik juga ada yang turun. Tapi ya begitulah. Bagaimanapun, jikalau maka, apabila, senantiasa dimanakah, jika maka jika, andaikata siapa dimana lah. Ah ngomong apa sih ini... Intinya mah, perjalanan itu kadang jadi nggak simple. Seperti saat ini, kami mohon maaf, di edisi ke-3 ini, kami nggak bisa memuat materi

pemrograman Python dan Bash. Untuk Python nggak bisa dimuat karena Kang Bram (Bramandityo Prabowo – Nyankodist Python) masih sibuk ngerjain Projectnya. Kemarin sih rencananya kang Bram mau ngisi lagi, tapi ternyata belum sempet juga. Adapun untuk pemrograman Bash juga nggak bisa dimuat karena kang Taufik (Taufik Sulaeman – Nyankodist Bash) lagi sakit. Doain ya moga cepet sembuh, supaya kedepannya bisa nulis lagi.

Akhirnya buat semua pembaca yang budiman, yang baik hati yang nggak alay, kami ucapkan selamat menikmati hidangan kami ini. Mohon maaf ya kalo ada kekurangan. Cemungudh eaa kakak...

***Coding is pulen bro!!!***



**Nyankodist Team**

Bila ada pertanyaan seputar Nyankod, atau mau bertanya tentang pemrograman kepada Nyankodist Team, atau hanya sekedar silaturahmi juga boleh, asalkan jangan spam, silahkan kontak kami ke email kami: [nyankod@gmail.com](mailto:nyankod@gmail.com).

Anda juga dapat berkomentar langsung seputar konten majalah di post artikel di situs Nyankod atau langsung menghubungi nyankodist pada kontak yang telah disediakan.

## Di dalam sini, ada..

Perl - [Operator-operator]	Halaman 9
PHP - Ooo.. PERATOR DAN KAWAN-KAWANNYA	Halaman 22
Hello #{RubyWorld} - Bagian Array dan Operator	Halaman 34
ActionScript {Operator}	Halaman 45
Operator di Bahasa C	Halaman 54
JavaScript - Variabel dan Operator	Halaman 60

**[Perl]**



Kresna Galuh D. Herlangga  
@kresnagaluh  
kresnagaluh@gmail.com  
<http://kresnagaluh.com>



# Operator-operator

Katakanlah kita punya sebuah variabel, misalkan namanya \$sembarang, dan variabel \$sembarang ini diisi dengan tipe data numbers bernilai 3. Menurut kamu, gimana caranya agar variabel \$sembarang tadi berubah nilainya menjadi 4 tapi tanpa mengganti nilai variabel secara langsung??

Gimana, bisa jawab pertanyaan di atas?

Oke, terlepas dari bisa kamu jawab atau nggak, kita lanjut aja dulu ya jou. Nanti di akhir pembahasan kamu pasti bakalan bisa jawab pertanyaan di atas dengan sangat yakin. So, mari kita mulai..... cemungudh eaaa....

Sebelumnya, ketemu lagi dengan saya dalam pembahasan Perl di Nyankod Magz. Dan nggak terasa, ternyata udah sampe pada edisi ketiga nih. Gimana masih inget dengan pembahasan-pembahasan di edisi sebelumnya? Masih inget kan? Masih inget dong, please....

Oke, buat ngereview, kemarin kita udah ngebahas tentang tipe data skalar dan variabel. Dan buat nguji apakah kamu udah paham atau belum tentang pembahasan sebelumnya, coba jawab 7 pertanyaan ini:

1. Apa itu tipe data?
2. Tipe data di Perl dibagi menjadi berapa?
3. Dalam tipe data skalar ada apa aja?
4. Apa yang kita gunakan untuk membatasi angka ribuan dalam Perl?
5. Apa bedanya tanda petik tunggal dan petik ganda dalam string?

6. Apa itu variabel?
7. Perlukah mendeklarasikan tipe datanya terlebih dahulu untuk membuat sebuah variabel?

Kalau kamu bisa ngejawab semua pertanyaan itu dengan baik, artinya kamu udah paham tentang pembahasan sebelumnya. Tapi kalau kamu nggak bisa jawab, silahkan pelajari lagi edisi sebelumnya. Karena, apa yang bakal kita bahas di edisi kali ini memerlukan pemahaman dari edisi sebelumnya.

Buat temen-temen yang baru gabung, dan kebingungan ini ngebahas apaan, mendingan coba download deh dua edisi sebelumnya, gratis kok. Pembahasanya menarik lagi.

Seperti kata pepatah, tak ada rotan sendal jepit pun jadi, so... mari kita mulai aja pembahasannya. Oh ya, jangan lupa berdoa ya, semoga terhindar dari godaan setan yang terkutuk.

## Apa sih Operator?

---

Operator adalah symbol yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi. Adapun suatu data yang dikenai operasi atau manipulasi oleh operator disebut dengan operand. Sebagai contoh yang paling sederhana operator yang paling sering kita temui di kehidupan sehari-hari adalah operator penjumlahan, pengurangan, perkalian dan pembagian. Oke, lihat ini:

$$2 + 3$$

Pada pernyataan di atas, 2 dan 3 disebut sebagai operand, adapun penjumlahan (+) disebut operator. Dengan operator penjumlahan tersebut maka

angka 2 dan 3 akan dimanipulasi sehingga menghasilkan angka 5. Ya, secara sederhana itulah operator.

## Operator Numerik

---

Operator numerik adalah operator yang dikenakan pada minimalnya sebuah numerik pada sebuah statement. Operator numerik digolongkan menjadi 4 kelompok, yaitu operator aritmatika, operator bitwise, operator perbandingan (persamaan dan pertidaksamaan) dan operator boolean.

### 1. Operator Aritmatika

Yang termasuk ke dalam operator aritmatika adalah penjumlahan (+), pengurangan (-), perkalian (\*), pembagian (/) dan sisa bagi (%). Penggunaannya sama seperti operator matematika biasa kok. Hanya saja untuk operator sisa bagi, berarti kita menghitung sisa dari hasil pembagian. Sebagai contoh misalnya sisa bagi dari  $5 \% 2 = 1$ . Jadi gini, kasusnya adalah bilangan yang bulat (nggak pake koma desimal), 5 dibagi 2 adalah 2 sedangkan sisanya adalah 1. Atau lebih gokil dan gilanya gini, Pak Ahmad punya jeruk 5 lembar. Dan dia punya anak 2 ekor. Dia mau ngasih jeruk-jeruk itu ke anaknya sama rata, tapi utuh. Nggak ada yg dibelah. Jadi masing-masing anaknya dapet 2 buah jeruk. So masih ada 1 jeruk lagi dong sisanya. Nah 1 itulah yg disebut dengan sisa bagi. Untuk lebih memahaminya, coba praktekin ini jou...

#### Program: operator\_aritmatika.pl

```
1 | #!/usr/bin/perl
2 | # nama program : backslash_petik.pl
3 |
```

```
4 | print "4 + 7 = ", 4 + 7, "\n";
5 | print "25 - 21 = ", 25 - 21, "\n";
6 | print "5 * 2 = ", 5 * 2, "\n";
7 | print "15 / 5 = ", 15 / 5, "\n";
8 | print "17 % 3 = ", 17 % 3, "\n";
```

Jalankan program tersebut!

```
$ perl operator_aritmatika.pl
4 + 7 = 11
25 - 21 = 4
5 * 2 = 10
15 / 5 = 3
17 % 3 = 1
$
```

Tuh kan, hasilnya sama seperti matematika waktu kita SD. Mengenai  $17 \% 3 = 1$ , tentu udah paham kan. Ya kasusnya mirip sama Pak Ahmad, 5 jeruk dan 2 anaknya tadi. Oh ya, kalau Pak Ahmad punya jeruk lagi dan masih ada sisanya, boleh nanti buat Anda kok. Yang penting minta baik-baik ya.... Hahahaha.

Nah sekarang coba tulislah program ini:

### **Program: prioritas\_operator.pl**

```
1 | #!/usr/bin/perl
2 | # nama program : prioritas_operator.pl
3 |
4 | print 4 + 7 * 3, "\n";
```

Sebelum menjalankan program tersebut, coba tebak dulu kira-kira berapa hasilnya, apakah 33 atau 25? Bila Anda sudah mempunyai tebakannya, maka jalankan program tersebut, lalu bandingkan hasil tebakan Anda dengan keluaran yang ditampilkan program. Samakah hasilnya dengan tebakan Anda? Mengapa demikian?

Yups hasil yang keluar di layar adalah 25, dan bukan 33. Jadi yang pertama dilakukan adalah proses perkalian terlebih dahulu walaupun berada di sebelah kanan. Hal itu terkait masalah prioritas operator. Perkalian memiliki prioritas yang lebih tinggi dari penjumlahan.

Kres, sepertinya ada yang kelewat deh.... ng.. apa itu? Perpangkatan. oh iya, di Perl juga mengenal perpangkatan sama seperti di matematika biasa, hanya penulisannya saja yang berbeda. Untuk perpangkatan  $5^2$  kita bisa tuliskan  $5^{**2}$ , begitu juga  $6^4$ , kita bisa tuliskan  $6^{**4}$ . jadi perpangkatan di Perl menggunakan simbol  $**$ . Adapun proses perhitungannya sama saja dengan perpangkatan biasa. Jadi kalau  $6^{**4}$ , itu artinya  $6*6*6*6 = 1296$ .

## 2. Operator Bitwise

Operator bitwise adalah operator yang akan memanipulasi bilangan numerik secara bit per bit. Oke, biar gampang ikuti aja dulu ya, ntr juga paham sendiri.

Pertama-tama, perhatikan beberapa point ini:

- 0 bila ditulis dalam bilangan biner adalah 0 juga. bila ditulis dalam bilangan biner 8 bit maka menjadi 00000000. Dan bila ditulis dalam bilangan biner 32 bit maka menjadi 00000000000000000000000000000000.
- 85 bila ditulis dalam bilangan biner adalah 01010101, so bila ditulis dalam biner 32 bit maka menjadi 00000000000000000000000001010101.
- 204 bila ditulis dalam bilangan biner adalah 11001100 dan bila ditulisnya dalam bilangan biner 32 bit maka menjadi 000000000000000000000000011001100.





Maka keluarannya adalah

```
$ perl operator_bitwise.pl
51 AND 85 = 17
204 OR 85 = 221
204 XOR 170 = 102
NOT 85 = 4294967210
$
```

### 3. Operator Perbandingan

Coba cek ini gan!

==	Sama dengan
!=	Tidak sama dengan
>	Lebih dari
<	Kurang dari
>=	Lebih dari atau sama dengan
<=	Kurang dari atau sama dengan

Dalam operator perbandingan apabila dua bilangan numerik dibandingkan dan hasilnya adalah benar, maka keluarannya adalah 1, namun jika salah maka keluarannya adalah string kosong "". Untuk lebih jelasnya cobalah jalankan program di bawah ini.

#### Program: operator\_perbandingan.pl

```
1 #!/usr/bin/perl
2 # nama program : operator_perbandingan.pl
3
4 print "5 == 2? ", 5 == 2, "\n";
```

```
5 print "3 == 3? ", 3 == 3, "\n";
6 print "4 != 9? ", 4 != 9, "\n";
7 print "2 != 2? ", 2 != 2, "\n";
8 print "4 > 8? ", 4 > 8, "\n";
9 print "5 > 3? ", 5 > 3, "\n";
10 print "9 < 8? ", 9 < 8, "\n";
11 print "1 < 5? ", 1 < 5, "\n";
12 print "4 >= 4? ", 4 >= 4, "\n";
13 print "7 >= 13? ", 7 >= 13, "\n";
14 print "6 <= 9? ", 6 <= 9, "\n";
15 print "5 <= 5? ", 5 <= 5, "\n";
```

Outputnya:

```
$ perl operator_perbandingan.pl
5 == 2?
3 == 3? 1
4 != 9? 1
2 != 2?
4 > 8?
5 > 3? 1
9 < 8?
1 < 5? 1
4 >= 4? 1
7 >= 13?
6 <= 9? 1
5 <= 5? 1
$
```

Sebenarnya sih masih ada satu lagi operator perbandingan yaitu `<=>`. Operator ini beda dari yang lainnya. Bila operator yang lain bila bernilai benar maka outputnya adalah 1, dan bila salah string kosong "". Nah dalam operator ini jika bilangan yg ada di sebelah kiri lebih besar maka keluarannya adalah 1, jika yang kiri

sama dengan yang kanan maka keluarannya adalah 0, dan jika yang kiri lebih kecil dari yang kanan maka keluarannya adalah -1. Nggak percaya? Silahkan dicoba sendiri.

#### 4. Operator Boolean

Operator ini berguna untuk memkombinasikan dua buah statment atau lebih. Adapun untuk caranya mirip dengan operator bitwise (benar dan salahnya, jika 1 AND 1 maka hasilnya 1, tapi jika 1 AND 0 hasilnya adalah 0), so buat kejelasannya prosesnya cek aja tabelnya. Hanya saja untuk lambang yang digunakan berbeda sedikit. Untuk AND, disini kita menggunakan &&, untuk OR kita menggunakan || dan untuk NOT kita menggunakan !. Buat lebih jelas, silahkan tulis kode berikut ini kemudian jalankan.

##### Program: operator\_boolean.pl

```
1  #!/usr/bin/perl
2  # nama program : operator_boolean.pl
3
4  print "6 > 3 && 2 > 4 : ", 6 > 3 && 2 > 4, "\n";
5  print "6 > 3 || 2 > 4 : ", 6 > 3 || 2 > 4, "\n";
6  print "! (6 < 7) : ", !(6 < 7), "\n";
7  print "! (6 > 7) : ", !(6 > 7), "\n";
```

Outputnya:

```
$ perl operator_boolean.pl
6 > 3 && 2 > 4 :
6 > 3 || 2 > 4 : 1
!(6 < 7) :
!(6 > 7) : 1
$
```

Dalam AND (&&) bila salah satu pernyataan ada yang bernilai FALSE baik itu di kiri atau di kanan maka hasilnya tetap FALSE. Untuk OR (||), maka akan bernilai TRUE karena salah satu dari statment khususnya yang kiri (6 > 3) bernilai TRUE. Baris selanjutnya (6 < 7) adalah pernyataan yang bernilai TRUE, namun bila diberikan NOT di depannya (!), maka akan bernilai FALSE. Bergitulah sebaliknya untuk baris terakhir.

## Operator String

---

Nggak cuma numerik yang bisa dikenakan operasi oleh operator, tapi string pun bisa, tentunya dengan operasi yang berbeda. Baiklah kita mulai dari yang paling sederhana yaitu menggabungkan dua buah string atau lebih, yaitu dengan menggunakan operator titik (.) atau koma (,). Saya rasa operasi ini sering kita praktekan. Contohnya seperti ini.

```
1 | print "pertama ", "kedua ", "string ketiga ", "tulisan terakhir  
2 | banget", "\n";
```

nah, perintah tadi juga bisa ditulis seperti ini:

```
1 | print "pertama " . "kedua " . "string ketiga " . "tulisan terakhir  
2 | banget" . "\n";
```

Output dari perintah di atas tentu kita udah tau semuanya, yaitu sebuah kalimat:

```
pertama kedua string ketiga tulisan terakhir banget
```

Terkadang yang kita butuhin nggak hanya menggabungkan string dengan string lainnya, tapi juga menggabungkan string dengan operasi numerik dan variabel. Untungnya di Perl itu disediakan, hehehe. Caranya masih sama kayak tadi, bahkan ini sering banget kita praktekin. Nih contohnya:

### Program: penggabungan\_string.pl

```
1 |#!/usr/bin/perl
2 |# nama program : penggabungan_string.pl
3 |
4 |$nama = "nyankoder";
5 |print "umur saya adalah " . 12*2 . " tahun" . "\n";
6 |print "saya seorang " . $nama . "\n";
```

Outputnya:

```
$ perl penggabungan_string.pl
umur saya 24 tahun
saya seorang nyankoder
$
```

Ada rotan ada kayu, biarin aja dah, ngapain repot sama rotan dan kayu. Jadi gini ceritanya, suatu saat waktu kamu masih SD, kamu terlambat sekolah, tiba-tiba ibu guru menghukum kamu dengan menyuruhmu buat nulis kalimat "saya berjanji tidak akan terlambat lagi" sebanyak 1000 kali. Nah kalo begitu gimana coba? Untung aja Perl punya solusi. Kamu cuma hanya menuliskan beberapa baris program saja, maka seribu kalimat langsung jadi. So nggak usah cape-cape lagi deh, dalam waktu kurang dari 1 menit udah jadi 1000 kalimat. Bahkan kalau ibu gurunya mau 1 juta kalimat juga nggak apa-apa. hehehehe. Nggak percaya? Coba jalanin deh program ini nih.

### Program: operator\_pengulangan.pl

```
1 |#!/usr/bin/perl
2 |# nama program : operator_pengulangan.pl
3 |
4 |print "saya berjanji tidak akan terlambat lagi \n" x 1000;
```

Outputnya:

```
$ perl penggabungan_string.pl
saya berjanji tidak akan terlambat lagi
... pokoknya sampe 1000 kali
$
```

Gimana bro pembahasanya kita kali ini?? Bisa dimengerti kan?? Oh ya, mengenai pertanyaan saya di awal mengenai bagaimana sebuah variabel \$sembarang bernilai 3 bisa menjadi 4 sudah bisa dijawab kan?? Oke, mungkin untuk pembahasan kita kali ini cukup sampe di sini dulu. Sebenarnya ada beberapa hal lagi terkait masalah string dan operator yang perlu kamu ketahui lagi, tapi sengaja saya siapkan itu buat edisi selanjutnya aja deh... Oke, sampe sini dulu bro... Tetap semangat ya belajar Perl.

*Coding is pulen bro...!!*

**[PHP]**



Ahmad Oriza Sahputra  
@oriza\_sahputra  
ahmadoriza@gmail.com  
<http://orizasahputra.blogspot.com>



# Ooo.. PERATOR DAN KAWAN-KAWANNYA

Salam Nyankoders!!!

*Assalamualaikum wr.wb..* Selamat pagi, siang, sore, malam untuk nyankoders sekalian (d disesuaikan aja bacanya jam berapa ya.. hehe). Telah lame ta' bejumpe, penulis harap nyankoders sekalian sudah paham bahasan artikel pada edisi kedua, yang belum tahu edisi sebelumnya "*cucian de lo..*", hehe.. becanda kok, maksudnya yang belum tempe ya download lah di <http://nyankod.blogspot.com> masa harus dikasi tahu!! Males begete!! Sekali lagi becanda kok nyankoders.. yang tadi hanya artikel pembuka saja, habisnya bingung mau bicara apa diawal. Jika melihat artikel Mas Kresna Krenyes, Kang Singgih, Bung Tarom Tarompet, dll suka ngiri juga, diawal menarik juga pembuka artikelnya. Haha.. kok jadi curhat. Okeh.. langsung saja ke ranah yang lebih tiga riu. Bacanya Chumungud ea kakek.. (maaf Bung Kresna Krenyes dipinjem kata-katanya).

Artikel pada edisi ini akan membahas operator (jenis-jenis & contoh) dan kawan-kawannya. Maksud dari kawan-kawannya disini adalah istilah yang terkait dengan operator yaitu ekspresi, operand dan konstanta.

## **Definisi**

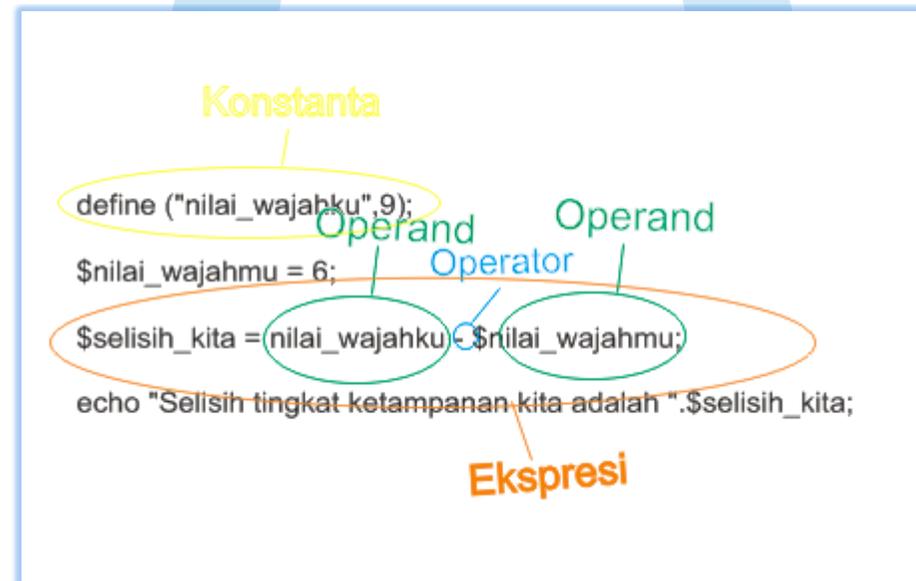
*"Ekspresi adalah suatu bentuk yang menghasilkan suatu nilai, dalam bentuk yang sederhana ekspresi dapat berupa konstanta atau variabel, dalam bentuk yang lebih kompleks suatu ekspresi dapat melibatkan suatu operand dan operator"* (Kasiman Peranginangin, p50).

"Operator adalah sesuatu (simbol atau karakter) yang digunakan untuk mengoperasikan suatu operand sehingga menghasilkan suatu nilai/hasil".

"Operand adalah suatu nilai yang dioperasikan operator dalam suatu ekspresi tertentu".

"Konstanta adalah suatu variabel yang memiliki nilai tetap karena inisialisasi nilai diawal pemberiannya tidak akan dirubah".

Untuk lebih memahami penjelasan teoritis diatas silahkan perhatikan gambar berikut :



Pada gambar terlihat contoh dari masing-masing istilah yang pada sebenarnya nyankoders sekalian telah menggunakannya dalam latihan coding di edisi kedua. Penggunaan konstanta, variabel, operator, operand, dsb akan terus dimanfaatkan

dalam kegiatan programming terlepas dari bahasa pemrograman apapun itu. Hanya dalam penulisannya akan ada sedikit perbedaan disesuaikan dengan bahasa pemrograman yang ada. Istilah teknis seperti diatas sudah seharusnya dipahami oleh programmer pemula :D.

## Lebih Dalam Lagi Tentang Operator

---

Operator dalam bahasa PHP dapat dibedakan dalam berbagai jenis dan mempunyai penggunaan yang berbeda-beda. Penggunaan yang berbeda-beda dan terdiri dari berbagai jenis dimiliki operator dalam bahasa PHP (kayanya ada yg aneh.. :D). Okeh.. mari kita pelajari masing-masing operator dalam bahasa PHP, siapkan editor masing-masing dan coba ketik source code latihan dibawah ini. Learning by doing akan lebih memudahkan kita untuk belajar :D, oia.. yang mau download source codenya bisa didownload di <http://www.mediafire.com/?uoddj62x3716xf7>.

### 1. Operator Aritmatika

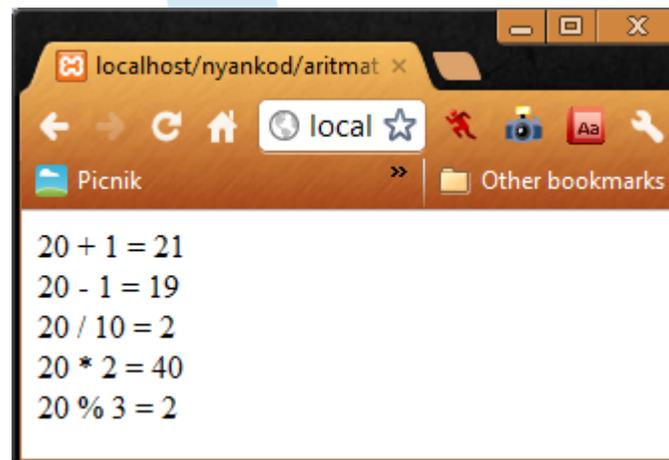
Jika anda ingin melakukan perhitungan yang bersifat matematis maka gunakanlah operator aritmatika.

Operator	Keterangan	Contoh
+	Penjumlahan	$\$x = 1 + 1$
-	Pengurangan	$\$x = 3 - 1$
*	Perkalian	$\$x = 1 * 1$
/	Pembagian	$\$x = 1 / 1$
%	Sisa Bagi	$\$x = 20 \% 3$

## Program: aritmatika\_op.php

```
1 <?
2 // nama program : aritmatika_op.php
3
4 $a = 20 + 1;
5 $b = 20 - 1;
6 $c = 20 / 10;
7 $d = 20 * 2;
8 $e = 20 % 3;
9
10 echo "20 + 1 = ".$a."<br>";
11 echo "20 - 1 = ".$b."<br>";
12 echo "20 / 10 = ".$c."<br>";
13 echo "20 * 2 = ".$d."<br>";
14 echo "20 % 3 = ".$e;
15 ?>
```

Output program tersebut apabila dijalankan pada browser akan tampak seperti gambar berikut:



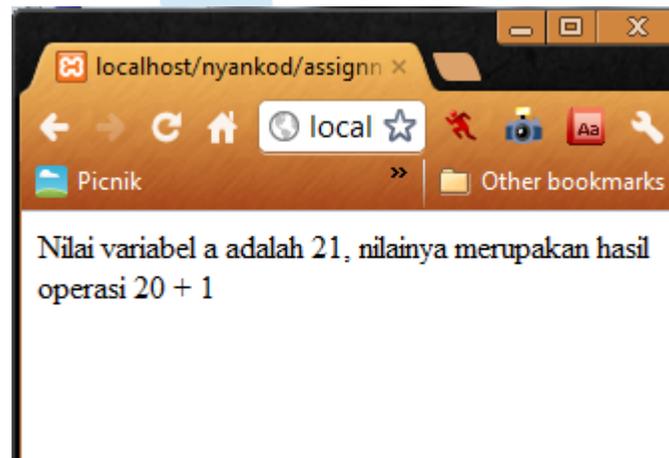
## 2. Operator Assignment

Operator (=) / Sama dengan, operator yang memberikan nilai kepada suatu variabel. Secara teknis operator ini memberikan nilai hasil operasi operand sebelah kanan untuk operand sebelah kiri. Operator ini sudah sering kita gunakan.

### Program: assignment\_op.php

```
1 <?
2 // nama program : assignment_op.php
3
4 $a = 20 + 1;
5 echo "Nilai variabel a adalah ".$a.", nilainya merupakan hasil
6 operasi 20 + 1";
7 ?>
```

Output :



### 3. Operator Relasi

Operator ini digunakan untuk membandingkan ekspresi ataupun nilai. Hasil yang ditampilkan adalah TRUE (1) atau FALSE (0).

#### Program: relasi\_op.php

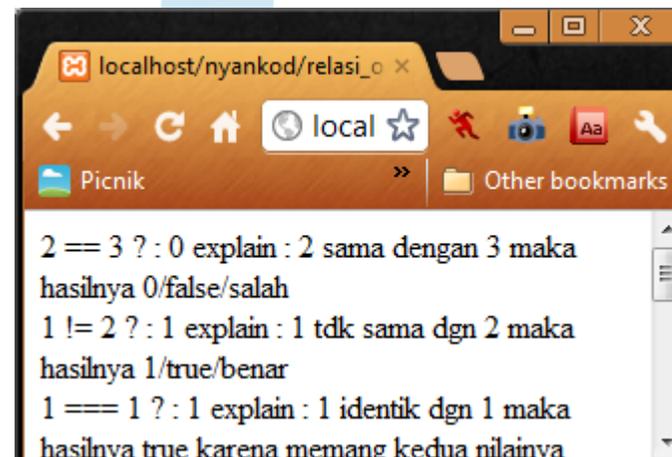
```
1 <?
2 // nama program : relasi_op.php
3
4 /*relasi sama dengan ( == )*/
5 printf ("2 == 3 ? : %d",2==3);
6 echo " explain : 2 sama dengan 3 maka hasilnya 0/false/salah<br>";
7
8 /*relasi tidak sama dengan ( != / <> )*/
9 printf ("1 != 2 ? : %d",1!=2);
10 echo " explain : 1 tdk sama dgn 2 maka hasilnya 1/true/benar<br>";
11
12 /*relasi identik ( === )*/
13 printf ("1 === 1 ? : %d",1===1);
14 echo " explain : 1 identik dgn 1 maka hasilnya true karena memang
15 kedua nilainya identik<br>";
16
17 /*relasi tidak identik ( !== )*/
18 printf ("1 !== 2 ? : %d",1!==2);
19 echo " explain : 1 tdk identik dgn 2 maka hasilnya true karena
20 memang kedua nilainya tdk identik<br>";
21
22 /*relasi lebih besar ( > )*/
23 printf ("1 > 2 ? : %d",1>2);
24 echo " explain : 1 lbh besar dari 2 maka hasilnya false karena 1
25 tdk lbh besar dari 2<br>";
26
27 /*relasi lebih kecil ( < )*/
```

```

28 printf ("4 < 6 ? : %d",4<6);
29 echo " explain : 4 lbh kecil dari 6 maka hasilnya true<br>";
30
31 /*relasi lebih besar atau sama dengan ( >= )*/
32 printf ("2 >= 1 ? : %d",2>=1);
33 echo " explain : 2 lbh besar atau sama dengan 1 maka hasilnya
34 true<br>";
35
36 /*relasi lebih kecil atau sama dengan ( <= )*/
37 printf ("1 <= 3 ? : %d",1<=3);
38 echo " explain : 1 lbh kcl atau sama dengan 3 maka hasilnya
39 true<br>";
40 ?>

```

Output :



#### 4. Operator Increment dan Decrement

Operator ini juga bersifat matematis. Tujuan dari operator ini adalah menyederhanakan ekspresi operator penambahan dan pengurangan dengan peningkatan atau penurunan satu nilai. Coba perhatikan program dibawah ini, operasi penambahan maupun pengurangan dibawah memiliki dua cara yang berbeda dengan hasil yang sama.

##### Program: inc\_dec.php

```
1 <?
2 // nama program : inc_dec.php
3
4 $a = 3;
5 //increment, penambahan satu nilai
6 $b = $a + 1;
7 $c = ++$a;//operasi increment
8 //decrement, penurunan satu nilai
9 $d = $a - 1;
10 $e = --$a;//operasi decrement
11 echo "Nilai b adalah $b sama dengan nilai c yaitu $c <br>";
12 echo "Nilai d adalah $d sama dengan nilai e yaitu $e";
13 ?>
```



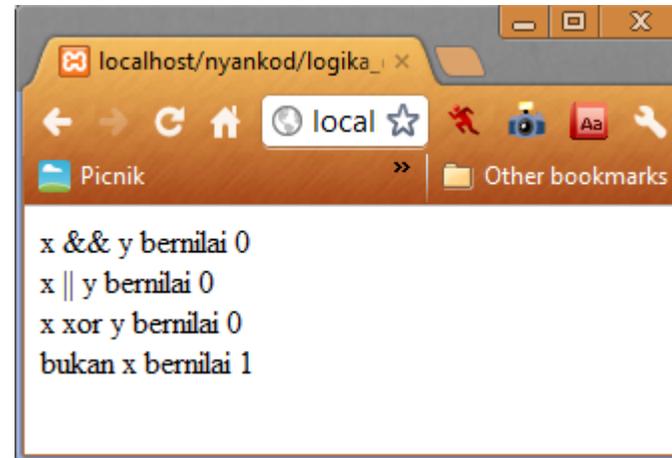
## 5. Operator Logika

Operator ini digunakan untuk operasi logika yang menghasilkan nilai TRUE atau FALSE.

### Program: logika\_op.php

```
1 <?
2 // nama program : logika_op.php
3
4 $x = 0;
5 $y = 1;
6 $a = $x and $y;
7 $b = $x or $y;
8 $c = $x xor $y;
9 $d = !$x;
10
11 /*logika 'dan' ( and / && )*/
12 echo "x && y bernilai ".$a."<br>";
13
14 /*logika 'atau' ( or / || )*/
15 echo "x || y bernilai ".$b."<br>";
16
17 /*logika 'atau eksklusif' ( xor )*/
18 echo "x xor y bernilai ".$c."<br>";
19
20 /*logika 'bukan' ( ! )*/
21 echo "bukan x bernilai ".$d;
22 ?>
```

Output :



A screenshot of a web browser window. The address bar shows 'localhost/nyankod/logika\_'. The browser interface includes navigation buttons (back, forward, refresh, home), a search bar with 'local', and a bookmarks bar with 'Picnik' and 'Other bookmarks'. The main content area displays the following text:

```
x && y bernilai 0  
x || y bernilai 0  
x xor y bernilai 0  
bukan x bernilai 1
```

Yup.. gimana nyankoders, gampang kan??. Mudah-mudahan penjelasan dan latihan di edisi ketiga ini bermanfaat untuk nyakoders sekalian. Apalagi kalo yang baca baru memulai PHP atau baru belajar pemrograman, mudah-mudahan tambah chumungud belajarnya :D. Oia nyankoders.. sebenarnya dalam edisi ini penulis mau menjelaskan juga tipe variabel objek atau pemrograman berorientasi objek (pbo) seperti yang dijanjikan di edisi sebelumnya. Tapi penulis berubah pikiran (lho kok.. xixi). Penulis rasa penjelasan pbo terlalu dini untuk disampaikan karena materi tentang fungsi dsb yang berhubungan dengan pbo belum tersampaikan, oleh karena itu penjelasan pbo akan disampaikan setelah materinya cukup.

***Keep Learning and Be The Best***

# [Ruby]



Muhammad Singgih Zulfikar Anshori  
@hirokakaoshi  
m.singgih.za@gmail.com  
<http://mszacompany.wordpress.com>

# Hello #{RubyWorld} – Bagian Array dan Operator



Owh iya, saya minta maaf di edisi kemarin ada beberapa kesalahan penulisan seperti penjelasan bilangan pecahan harusnya angka pecahan, dan juga ada bilangan cacah harusnya angka pecahan juga. Moho maaf sebesar besarnya ( $\cong \nabla \leq$ )y .

Lets Rock en Roll...

Yo berjumpa lagi dengan nyankods bagian Ruby, hha gimana yang edisi kemarin ada yang ditanyakan tidak? Soalnya ga ada yang me-email saya sih kalau ada pasti saya jawab dan masukan ke edisi nyankods selanjutnya...

Tapi gapapa deh mungkin sudah cukup untuk dimengerti ketika belajar dari panduan Ruby yang saya buat ini, perlahan namun pasti...hheu

sekarang kita masuk ke lanjutan materi kemarin yaitu tentang operator dan number, namun ditambah dengan Array nih sekarang. Masih inget ga apa yang ada dalam tabel operator di edisi nyankods kemarin, hayo ?? bagi yang punya silahkan dibuka kembali kalau lupa edisi kemarin, dan bagi yang belum punya silahkan unduh di <http://nyankods.blogspot.com>.

Mari kita review apa aja sih yang udah dipelajari di edisi kemarin, diantaranya sebagai berikut :

1. Kamu sudah mengetahui tipe angka di Ruby yaitu integer dan float.
2. Kamu sudah tau apa aja operator yang ada di Ruby, ada di tabel itu loh.
3. Kamu sudah tau sekarang bagaimana untuk mengatasi penggunaan (++) atau (-) yang biasanya ada di bahasa pemograman lain di Ruby.
4. Kamu sudah mengerti dan jago menggunakan += -= \*= /= %= .

Nah sudah ingat sekarang? Apa saja yang sudah dipelajari di edisi kemarin.

## Level Up

Seperti biasa sebelum berlatih Ruby, persiapkan bahan pokok berikut ini :

1. Buka Folder RubyDojo.
2. Buka Editor teks kesayangan kamu.
3. Buka Ruby editor interaktif (irb) di terminal | cmd.

Edisi kemarin sudah dijelaskan mengenai angka bulat(integer) dan angka pecahan (float), dan juga sifat keduanya. Di Level ini kamu akan belajar mengenai Array (\*wow apaan Tuh Qaqaa??) hohoho...

Nah Loh??? Ngerti ga maksudnya apa??? .... (ò.ò)

Di Ruby, Array menggunakan bracket atau tanda kurung siku ( [] ) sebagai pembungkus konten isinya. \*ingat lagi tabel operator di edisi 2 kemarin. Supaya lebih jelas saya kasih contoh ya, kamu juga coba praktekan di irb ok..



Array atau dalam bahasa indonesianya disebut dengan larik di bahasa pemrograman konvensional seperti C atau java merupakan sekumpulan variabel bertipe sama yang diacu dengan nama yang sama, masing-masing elemen Array dapat diakses melalui indeks nya.

Namun dalam Ruby, Array adalah sekumpulan variabel yang dapat berbeda tipe dan diacu dengan nama yang sama.

```
1.9.2p290 :001 > akuArray = [3, "tiga", 2.9999]
=> [3, "tiga", 2.9999]
1.9.2p290 :003 > akuArray.class
=> Array
1.9.2p290 :004 > akuArray.length
=> 3
```

Kenapa hal ini bisa terjadi!!!! oh tuhan!!!

Array di Ruby koq bisa ya bertipe berbeda padahal dalam satu bungkusan yang sama, kayak parcel aja isinya macem-macem wkwkwk...

Begini ceritanya, Ruby merupakan bahasa pemrograman yang sudah menerapkan Object Oriented Programming, jadi sebenarnya yang kita kenal sebagai



Perlu dipahami, dalam bahasa pemrograman apapun untuk pemanggilan isi Array dimulai dengan konten indeks ke 0 bukan dari indeks 1. Seperti contoh di atas, isi indeks 0 yaitu integer 3, indeks 1 isinya string "tiga", dan indeks 2 isinya float 2.99, namun ketika dipanggil indeks 3 maka isinya adalah nil karena indeks ketiga tidak mempunyai isi begitu pun seterusnya.

variabel merupakan suatu objek, seperti pada penjelasan di edisi pertama tentang pengenalan ruby. Ternyata objek bersifat flexibel sesuai dengan kebutuhan kita, jadi jangan heran kalau kedepannya tipe data bisa berubah rubah di satu pemuat/variabel atau tepatnya disini disebut objek.

Oke?? \*silahkan baca penjelasan di paragraf atas 3x berturut-turut supaya paham..wkwkwkwk. Contoh tadi juga ada perintah `akuArray.class` dan `akuArray.length`, perintah pertama untuk mengetahui tipe dari objek `akuArray` dan perintah kedua untuk mengetahui panjang Objek `akuArray`.

Selanjutnya coba kita tampilkan isi dari Array yang sudah dibuat, bagaimana cara kita meng-Summon nya?? (\*summon adalah bahasa jepang yang artinya memanggil, efek keseringan baca komik jepang).

```
irb(main):001:0> akuArray[0]
=> 3
irb(main):002:0> akuArray[1]
=> "tiga"
irb(main):003:0> akuArray[2]
=> 2.9999
irb(main):004:0> akuArray[3]
=> nil
```

Selain dengan cara di atas, pemanggilan indeks dapat dilakukan dengan menggunakan angka minus atau negatif, contoh nih:

```
irb(main):005:0> akuArray[-3]
=> 3
irb(main):006:0> akuArray[-2]
=> "tiga"
irb(main):007:0> akuArray[-1]
=> 2.9999
```

Selain itu, pemanggilan indeks di Array dapat dilakukan dari indeks ke berapa kemudian berapa banyak yang akan ditampilkan, dengan format [indeks berapa, jumlah yang akan ditampilkan].

```
irb(main):008:0>ArrayNih=  
["satu",2,"tiga",4,"lima",6,"tujuh"]  
=> ["satu", 2, "tiga", 4, "lima", 6, "tujuh"]  
irb(main):009:0> ArrayNih[1,3]  
=> [2, "tiga", 4]  
irb(main):010:0> ArrayNih[-4, 2]  
=> [4, "lima"]
```

See, ArrayNih[1,3] menampilkan isi dari indeks ke 1 sampai 3 indeks kedepannya, juga ArrayNih[-4,2] menampilkan isi dari indeks ke -4 sampai 2 indeks kedepannya.

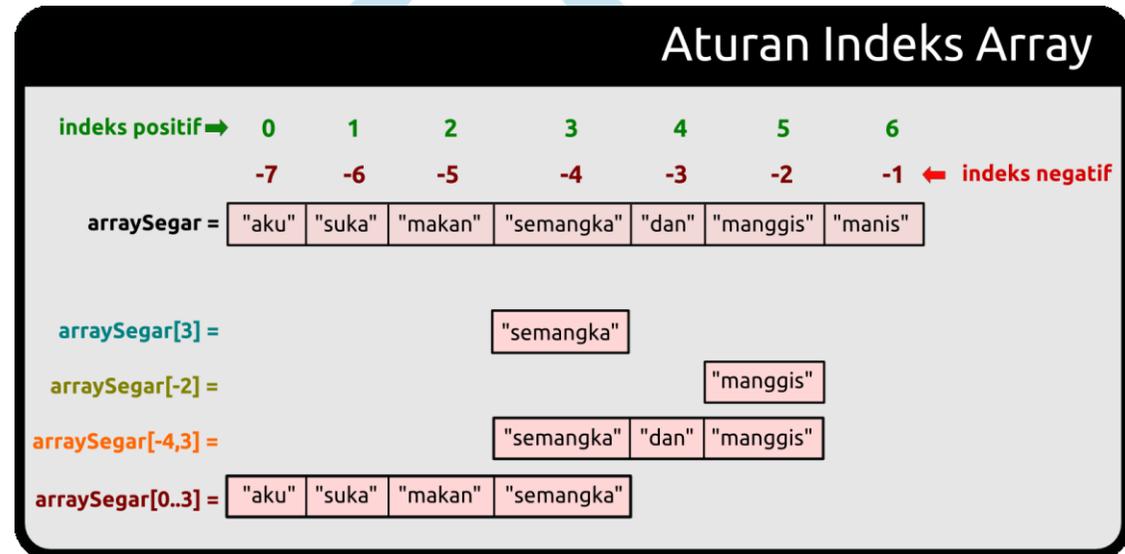
```
[+++=====___=====+++] I Am A BOTS!!!!
```

kurung sikut ini unik banget ya, dan ternyata pemanggilannya di Array ada lagi cara lainnya, yaitu dengan range atau jangkauan dari indeks berapa sampai ke indeks berapa, seperti kita mau print halaman dari 3 sampai 9. Begini caranya, [mulai..akhir] nah perintah ini yang dipakai untuk memanggil indeks Array.

```
irb(main):011:0> ArrayNih[1..5]  
=> [2, "tiga", 4, "lima", 6]  
irb(main):012:0> ArrayNih[0..3]  
=> ["satu", 2, "tiga", 4]  
irb(main):013:0> ArrayNih[4..-2]  
=> ["lima", 6]
```

Sip kan, paham dengan contoh di atas...silahkan kamu coba ya..

Apabila masih kesulitan memahami pemosisian indeks dan penghitungannya, coba deh perhatikan tabel dibawah supaya lebih paham cara membaca indeks Array di Ruby....



Baiklah, semoga dapat memperjelas tentang Array dengan tabel di atas.

Sudah jauh juga ternyata latihan ini, sekarang saatnya menguji kemampuan kamu sekalian saya break sebentar haha...

OK kerjakan soal berikut :

1. Buat Array bulan dengan isi 12 indeks, indeks pertama berisi jumlah hari dalam bulan januari, indeks kedua berisi jumlah hari dalam february dan terus sampai bulan desember.
2. Panggil isi dari bulan juni sampai oktober.
3. Ubah isi bulan february menjadi 29 (nah Loh susah).

## HERO state :: Extra Fokus

---

Siap siap yaa.. di sini harus lebih fokus lagi, yang tadinya 100% menjadi 200% OK.. Sebelumnya kita sudah mempelajari bagaimana cara membuat Array dan memanggilnya, tapi belum membahas mengenai bagaimana cara mengisi Array dan mendeklarasikan objek Array.. Ok, sekarang kita akan membahas itu...

Array merupakan sebuah kontainer yang dapat berisi tipe data apapun di Ruby karena dikenali sebagai objek, dan Array pula dapat ditambahkan jumlah indeks nya sesuai dengan keinginan kita. Apabila di pemrograman konvensional mungkin memerlukan List karena belum menerapkan OOP, ada istilah pointer dan sebagainya. Daripada bingung dengan apa itu list, sekarang perhatikan saja Array di Ruby OK...

```
irb(main):014:0> ArrayFlexy = Array.new
=> []
irb(main):015:0> ArrayFlexy.class
=> Array
irb(main):016:0> ArrayFlexy.length
=> 0
irb(main):017:0> ArrayFlexy[1] = "dua"
=> "dua"
irb(main):018:0> ArrayFlexy
=> [nil, "dua"]
irb(main):019:0> ArrayFlexy[0] = 1
=> 1
irb(main):020:0> ArrayFlexy
=> [1, "dua"]
```

Wowow, lihat itu...udah dicobain kan? Ada yang aneh ga??

Di atas saya membuat Objek Array baru dengan nama ArrayFlexy, nah yang aneh kenapa saya isi indeks 1 dahulu?? error donk, tapi ternyata tidak berlaku untuk

Array yang dikenali sebagai Objek. Ketika dipanggil ArrayFlexy, isi indeks pertama nil karena belum mempunyai isi, selanjutnya saya isi dengan angka 1 dan hasilnya berubah menjadi [1,"dua"].

Disinilah fleksibilitas dari deklarasi sebagai objek dalam OOP, memudahkan untuk pengoprasian dan tidak perlu repot lagi dengan bahasa pemrogramannya membuat kita lebih fokus ke kreasi algoritma saja selanjutnya...ohohoh

### 1. Penggunaan operator + - \* dalam Array :

```
irb(main):021:0> [1,2,3,4] + [ 3,4,5,6]
=> [1, 2, 3, 4, 3, 4, 5, 6]
irb(main):022:0> [1,2,2,3,4,4,4,5,6,7] - [2,3]
=> [1, 4, 4, 4, 5, 6, 7]
irb(main):023:0> [1,2,3,4] * 2
=> [1, 2, 3, 4, 1, 2, 3, 4]
```



Sebenarnya banyak sekali yang perlu dijabarkan dalam Array, namun berikut ini saya berikan beberapa contoh perintah yang sering digunakan atau kasus yang sering muncul.... silahkan disimak dan dicoba :

### 2. Penggunaan ( & | ) dalam Array :

```
irb(main):024:0> [1,2,3,4] & [4,5,6,7]
=> [4]
irb(main):025:0> [1,2,3,4] | [4,5,6,7]
=> [1, 2, 3, 4, 5, 6, 7]
```

### 3. Penggunaan sorting dan indexing dalam Array :

```
irb(main):026:0> a= ['a','c','o','f','d']
=> ["a", "c", "o", "f", "d"]
irb(main):027:0> a.index("c")
=> 1
irb(main):028:0> a.index('c')
```

```
=> 1
irb(main):029:0> a.sort
=> ["a", "c", "d", "f", "o"]
```

sort : perintah untuk menyusun isi Array secara hirarkis.  
Index: mencari indeks dari isi yang dimaksud.

#### 4. Penggunaan pop dan push :

Push merupakan perintah untuk menambahkan isi ke Array dimulai dari indeks terakhir pada Array.

```
irb(main):030:0> a
=> ["a", "c", "o", "f", "d"]
irb(main):031:0> a.pop
=> "d"
irb(main):031:0> a
=> ["a", "c", "o", "f"]
irb(main):032:0> a.push("d","e","g")
=> ["a", "c", "o", "f", "d", "e", "g"]
```

pop : perintah untuk menghapus indeks terakhir dalam Array.

Push : perintah untuk menambahkan isi ke Array dimulai dari indeks terakhir pada Array

#### 5. Penggunaan insert, delete, dan replace :

```
irb(main):033:0> a
=> ["a", "c", "o", "f", "d", "e", "g"]
irb(main):034:0> a.insert(2,"b")
=> ["a", "c", "b", "o", "f", "d", "e", "g"]
irb(main):035:0> a.insert(2,"b","c","d")
```

```

=> ["a", "c", "b", "c", "d", "b", "o", "f", "d", "e", "g"]
irb(main):036:0> a.replace(["a","i","u","e","o"])
=> ["a", "i", "u", "e", "o"]
irb(main):037:0> a
=> ["a", "i", "u", "e", "o"]
irb(main):038:0> a.delete("e")
=> "e"
irb(main):039:0> a
=> ["a", "i", "u", "o"]

```

insert : merupakan perintah untuk memasukan isi dimulai dari indeks yang ditentukan (indeks berapa, isinya).

Replace : perintah untuk mengganti isi keseluruhan Array.

Delete : perintah untuk menghapus isi tertentu dari seluruh indeks yang ada dalam Array.

## 6. Penggunaan to\_s dan uniq dan clear

```

irb(main):040:0> a
=> ["a", "i", "u", "o"]
irb(main):041:0> a.to_s
=> "aiuo"
irb(main):042:0> a = ["a","a","i","i","u","e","o","o"]
=> ["a", "a", "i", "i", "u", "e", "o", "o"]
irb(main):043:0> a.uniq
=> ["a", "i", "u", "e", "o"]
irb(main):044:0> a.clear
=> []

```

to\_s : perintah untuk mengubah seluruh indeks menjadi kesatuan dalam bentuk string.

Uniq : perintah untuk mencari isi yang unik apabila terdapat perulangan atau isi yang sama di seluruh indeks.

Clear : perintah untuk membersihkan isi Array.

Wadaw banyak amat euy, padahal ini baru beberapa fungsi dari sekian banyak fungsi yang digunakan dalam Array di Ruby.

Baiklah ini akhir dari Hero State..... Alhamdulillah (^\_^)9  
Sekarang semua kode yang telah kita pelajari silahkan buat 1 file ruby dengan nama EDISI3.rb . Kemudian simpan di Folder RubyDOJO.

Akhirnya, setelah lama-lama berlatih di RubyDojo, kamu sudah lolos naik ke level 3 ini. Apa aja nih yang sudah kita latih dalam level ini :

1. kamu sudah mengenal sifat-sifat Array di Ruby.
2. Kamu sudah bisa memanggil isi dari Array berdasarkan aturan indeks nya.
3. Kamu sudah bisa membuat objek Array dan mengisinya dengan leluasa karena Array di Ruby adalah Objek.
4. Kamu sudah mengetahui beberapa fungsi yang sering digunakan dalam Array dan sangat fungsional nantinya.

Array akan sering digunakan dalam tumpukan, antrian FIFO dan sebagainya yang berbau ngantri gituu, SEMANGAT YEAH.....

Salam akhir saya harap tulisan ini dapat bermanfaat bagi kamu dan lebih tertarik lagi untuk PDKT dengan Ruby, mohon maaf apabila ada kesalahan kata atau kurang berkenan.

Apabila ada pertanyaan dan sebagainya bisa menghubungi saya di kontak : <http://about.me/MuhammadSinggihZA>  
Salam, hirokakaoshi  
| Nyankodist Ruby.

# [ActionScript]



Tarom Apriyanto  
@tarompey  
tarompey@gmail.com



# Operator

Ketemu lagi sobat Nyankod (Nyankoder). Apa kabar??,...Baik donk...!!

Terimakasih untuk tetap setia mengikuti materi-materi yang diberikan oleh para Nyankodist. Ok sobat Nyankod, kali ini kita bakal melanjutkan pembelajaran kemarin, nah sekarang sobat Nyankod sudah kenal apa itu variabel bagaimana menggunakannya kan,..?? atau lupa,..??.....kalau begitu silahkan sobat Nyankod untuk membaca ulang dan pahami lagi, apabila masih kurang jelas sobat Nyankod bisa baca buku mengenai variabel atau sumber lainnya, karena kami (Nyankodist) tentunya tidak akan luput dari kekurangan, pada intinya kita meski rajin-rajin membaca supaya mudah mengerti.

Oke cukup Mukodimahnya, kita akan masuk materi selanjutnya yaitu OPERATOR.

Ada yang tahu apa itu OPERATOR anak-anak?? (malah kayak ngajar murid SD,..haahahaaha)

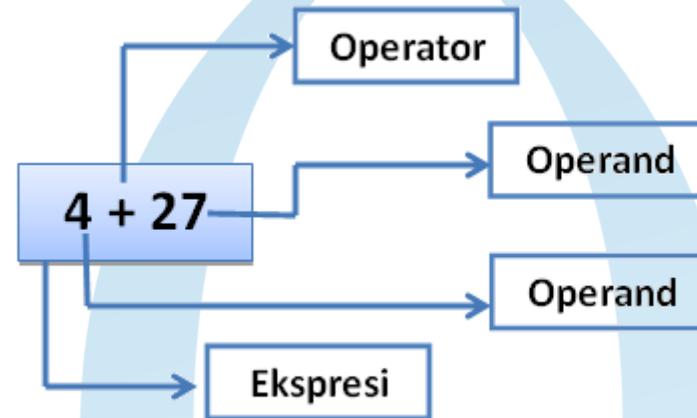
It was just Intro....Now, we will begin..!

## Mengenal Operator

---

Pada penjumlahan atau operasi seperti  $4 + 27$ , tanda  $+$  dinamakan operator. Operator berupa symbol yang digunakan untuk menyusun suatu ekspresi, dengan melibatkan satu atau beberapa operand, tergantung dari jenis operatornya. Pada

contoh di awal tadi yaitu  $4 + 27$ , ada dua operand yang terlibat, yaitu 4 dan 27. Adapun  $4 + 27$  sendiri dinamakan ekspresi, dalam hal ini ekspresi dapat digunakan untuk melakukan perhitungan atau bahkan perbandingan.



Berikut ini adalah beberapa jenis-jenis operator:

1. **Arithmetic Operator**, yaitu operator yang digunakan dalam perhitungan matematika. *Check this out.*

Simbol	Keterangan	Contoh
+	<i>Addition</i> , penambahan	skor = skor + 100;
-	<i>Subtraction</i> , pengurangan	skor = skor - 50;
*	<i>Multiplication</i> , perkalian	skor = skor * 2;
/	<i>Division</i> , Pembagian	skor = skor/2;
%	<i>Modulo</i> , sisa bagi hasil	skor = 27 % 7; Maka hasil Nilai adalah 6, karena sisa hasil bagi

		antara 27 dengan 7 adalah 6.
++	<i>Increment</i> , menambahkan 1	k++; k akan bertambah 1 nilainya. Sebagai contoh jika k asalnya 27, maka berubah menjadi 28. Perintah k++; sama hasilnya dengan k = k + 1;
--	<i>Decremen</i> , mengurangi 1	k--; k akan berkurang 1 nilainya. Sebagai contoh jika k asalnya 27, maka berubah menjadi 26. Perintah k--; sama hasilnya dengan k = k - 1;

2. **Assigment Operator** yaitu operator untuk mengisikan sebuah nilai kepada suatu variabel. *Check* lagi nih.

Simbol	Keterangan	Contoh
=	<i>Assigment</i> , memberikan nilai pada suatu variabel	skor = 300;
+=	<i>Addition assigment</i> , menambah suatu nilai pada sebuah variabel	Skor += 100; sama dengan skor = skor +100;
-=	<i>Subtraction assigment</i> , mengurangi suatu nilai pada sebuah variabel	Skor -= 50; sama dengan skor = skor - 50;
*=	<i>Multiplication</i>	Skor *= 2; sama dengan

	<i>assignment</i> , mengalikan suatu nilai pada sebuah variabel	skor = skor * 2;
/=	<i>Division assignment</i> , membagi suatu nilai pada sebuah variabel	Skor /= 2; sama dengan skor = skor / 2;
%=	<i>Modulo assignment</i> , mencari nilai modulo suatu variabel menggunakan suatu nilai tertentu	var p:Number = 27; p %= 7; maka nilai p adalah 6.

3. **Comparison operator**, yaitu operator untuk membandingkan dua variabel atau variabel dengan sebuah nilai yang kemudian keluarannya adalah nilai *true* dan *false*. *True* jika hasil pembandingan itu terpenuhi, *false* jika tidak. Lihat tabel nih,

Simbol	Keterangan	Contoh
==	<i>Equality</i> , memeriksa apakah bagian sebelah kiri memiliki nilai yang sama dengan bagian sebelah kanan.	If (totalskor == skor) {} Contoh lain: If (username = "Nyankoder") {}
===	<i>Strict equality</i> , memeriksa apakah dua variabel memiliki kesamaan nilai dan tipe data.	If (totalskor === skor) {}
!=	<i>Inequality</i> , memeriksa apakah bagian sebelah kiri memiliki nilai yang <b>tidak</b> sama dengan bagian sebelah kanan.	If (totalskor != skor) {} Contoh lain: If (username != "Nyankoder") {}
!==	<i>Strict inequality</i> , memeriksa apakah dua variabel berbeda nilai dan tipe data.	If (totalskor !== skor) {}

<	<i>Less than</i> , memeriksa apakah bagian sebelah kiri memiliki nilai lebih kecil daripada bagian sebelah kanan.	If (skorr < 27) {} Contoh lain: If (skor < totalskor) {}
<=	<i>Less or equal to</i> , memeriksa apakah bagian sebelah kiri memiliki nilai lebih kecil atau sama dengan bagian sebelah kanan.	If (skorr <= 27) {} Contoh lain: If (skor <= totalskor) {}
>	<i>Greater than</i> , memeriksa apakah bagian sebelah kiri memiliki nilai lebih besar daripada bagian sebelah kanan.	If (skorr > 27) {} Contoh lain: If (skor > totalskor) {}
>=	<i>Greater or equal to</i> , memeriksa apakah bagian sebelah kiri memiliki nilai lebih besar atau sama dengan bagian sebelah kanan.	If (skorr >= 27) {} Contoh lain: If (skor >= totalskor) {}

4. **Logical operator**, yaitu operator yang digunakan dalam operasi Boolean. Lihat tabel lagi,

Simbol	Keterangan	Contoh
!	NOT, membalik nilai dari suatu variabel Boolean.	If (akhir = "false" maka !akhir = "true") {}
&&	AND, operator yang berguna untuk memeriksa kebenaran semua bagian kondisi. Nilai kondisi akan <i>true</i> jika semua bagian kondisi tersebut bernilai	If ((UTS > 60 ) && (UAS>60)) { Status = "LULUS"; } Nilai keseluruhan kondisi akan <i>true</i> jika nilai UTS

	<i>true</i> . Namun jika ada salah satu atau lebih bagian kondisi <i>false</i> , maka nilai kondisi secara keseluruhan adalah <i>false</i> .	> 60 <b>dan</b> nilai UAS > 60.
	OR, logical operator yang akan memberikan nilai <i>true</i> jika minimal salah satu bagian kondisinya adalah <i>true</i> .	If ((UTS > 60 ) && (UAS>60)) { Status = "LULUS"; } Nilai keseluruhan kondisi akan <i>true</i> jika nilai UTS > 60 <b>atau</b> nilai UAS > 60.

## Coba Yuuuuuk....!!

Nah sekarang sobat Nyankod bisa praktekin dari beberapa contoh operator di atas untuk kita terapkan di ActionScript 2.0. Cobain Ini deh, untuk contoh penggunaannya :

1. Buat dokumen flash baru
2. Buka panel Action (nih kalau lupa buka panel action : Window > Action), terus ketik script di bawah ini :

```

1 // contoh satu
2 var skorku:Number = 0;
3 skorku = skorku + 1;
4 trace("contoh satu: " + skorku); // hasilnya 1

```

```
5 // contoh dua
6 var skore_lagi:Number = 1;
7 skore_lagi += 3;
8 trace("contoh dua: " + skore_lagi); // hasilnya 4
```

3. Pilih Control > Test Movie (ctrl + enter)  
~lihat hasilnya di panel display output~

### **Cobain lagi nih !!**

Scriptnya sobat Nyankod ganti dengan ini:

```
1 var angka:Number = 0;
2 angka++;
3 trace(angka); // hasilnya 1
4 angka++;
5 trace(angka); // hasilnya 2
```

Sekarang ganti dengan yang ini:

```
1 var i:Number;
2 for (i = 1; i < 10; i++) {
3   trace(i);
4 }
```

Coba ganti dengan yang ini:

```
1 var matika:Number;
2 matika = 2 + 4 * 3;
3 trace(matika); // hasilnya 14
```

Cobain yang satu ini juga:

```
1 | var mySum:Number;  
2 | mySum = (2 + 4) * 3;  
3 | trace(mySum); // hasilnya 18
```

Setelah itu, coba ganti dengan yang ini:

```
1 | var matika:Number;  
2 | var matika_lagi:Number;  
3 | matika = 2 * 4 * 3;  
4 | matika_lagi = (2 * 4) * 3;  
5 | trace(matika); // hasilnya 24  
6 | trace(matika_lagi); // hasilnya 24
```

Terakhir, cobain yang ini:

```
1 | trace(3 > 2 < 1); // hasilnya false  
2 | trace((3 > 2) < 1); // hasilnya false
```

Oke sobat Nyankod Cukup dulu yaa untuk sesi kali ini, nanti kita lanjut lagi materi selanjutnya di edisi-4 nyankodMagz. Buah semangka buah duku, semangat kaka udah dulu....!!

[C]



Ade Kurniawan  
@adekurniawan  
noadekur@yahoo.com



# Operator di Bahasa C

Operator atau tanda operasi adalah suatu tanda atau simbol yang digunakan untuk suatu operasi tertentu. Jika ingin ditetapkan nilai suatu variabel, selain memerlukan nama variabel itu, juga diperlukan operator tertentu (disebut assignment operators). Misalnya, tanda "=" pada deklarasi variabel `int i=22`.

## Jenis – jenis operator pada Bahasa C

---

### Operator Assigment

Operator Penugasan (Assignment operator) dalam bahasa C berupa tanda sama dengan ("="). Contoh :

```
1 | nilai = 80;  
2 | A = x * y;
```

Artinya : variable "nilai" diisi dengan 80 dan variable "A" diisi dengan hasil perkalian antara x dan y.

### Operator Aritmatika

*	untuk perkalian
---	-----------------

/	untuk pembagian
%	untuk sisa pembagian (modulus)
+	untuk penambahan
-	untuk pengurangan

Catatan : operator % digunakan untuk mencari sisa pembagian antara dua bilangan. Misalnya :

$$9 \% 2 = 1(9 : 2 = 8, \text{ sisa pembagian} = 1)$$

$$9 \% 3 = 0(9 : 3 = 3, \text{ sisa pembagian} = 0)$$

### **Operator Bitwise (>>, <<)**

Operator ini digunakan untuk memanipulasi nilai bit. Contoh:  $9 \gg 2$  hasilnya adalah 2. Bagaimana bisa begitu? Mari kita lihat pengoperasiannya dalam bilangan biner.

$$9 \Rightarrow 0000000000001001$$

$9 \gg 2$  maksudnya nilai biner pada angka 9 digeser 2 digit ke kanan.

2 digit 0 ditambahkan di sebelah kiri (00) $\Rightarrow 00000000000010(01) \Rightarrow$  2 digit sebelah kanan dihilangkan.

hasilnya:  $0000000000000010 \Rightarrow$  ini adalah angka biner dari 2

Jika kamu bingung, mungkin kamu perlu belajar cara mengkonversikan angka desimal ke biner dan sebaliknya.

### Operator Hubungan (>, <, >=, <=)

Operator ini berhubungan dengan penentuan nilai TRUE dan FALSE.

>	lebih besar dari
<	lebih kecil dari
>=	lebih besar dari atau sama dengan
<=	lebih kecil dari atau sama dengan

### Operator Persamaan dan pertidaksamaan (==, !=)

Operator ini juga berhubungan dengan penentuan nilai TRUE dan FALSE.

==	sama dengan
!=	tidak sama dengan

### Operator Logika (&&, ||)

Operator ini digunakan dalam pengkondisian / if.

&&	And (dan)
	Or (atau)

Berikut Disajikan Operator dalam bahasa C yang lebih lengkap

Operator	Arti	Contoh
()	memanggil fungsi	printf()

[]	elemen array (deret)	int y [11]
.	anggota struktur	ed.jumlah = 2200
!	NOT	
++	inkremen(tambah satu satu)	i++;
-	dekremen(kurang satu satu)	i-;
&	address dari ...	scanf("%c",&x);
*	isi dari ...	*kata
*	perkalian	x=y*z;
/	pembagian	x=y/z;
%	modulo (sisa hasil bagi)	a=y % z;
+	tambah	X= Z+ Y;
-	kurang	Z = X - Y;
<	lebih kecil daripada	A < 30;
>	lebih besar daripada	B > 24;
<=	lebih kecil atau sama dengan	C <= 15;
>=	lebih besar atau sama dengan	D >= 29;
==	kesamaan	X == 20;
!=	ketidaksamaan	K != 4;
=	menetapkan nilai	x=3;

## Contoh penulisan operator dalam bahasa c :

### Operator Aritmatika

```

1 | # include <stdio.h>
2 | main()
3 | {
4 |   int x,y ;
5 |   float z;

```

```

6
7     x = 7;
8     y = 3;
9
10    z = x/y;
11    printf("nilai z = = %f",z);
12    }

```

### Operator Bitwise

```

1     # include <stdio.h>
2     main()
3     {
4     int x=0x2d; /*bernilai 45 dalam desimal */
5     int y=0x1b; /*bernilai 27 dalam desimal */
6
7     printf("%x & %x hasilnya adalah %x\n",x,y,x&y);
8     printf("%x ^ %x hasilnya adalah %x \n",x,y,x^y);
9     }

```

### Operator Hubungan

```

1     # include <stdio.h>
2
3     main()
4     {
5     int a = 7, b = 9;
6
7     printf(" %d < %d hasilnya adalah %d\n",a,b,a<b);
8     printf(" %d == %d hasilnya adalah %d\n",a,b,a==b);
9     printf(" %d != %d hasilnya adalah %d\n",a,b,a!=b);
10
11    }

```

# [JavaScript]



Toni Haryanto  
@yllumi  
toha.samba@gmail.com  
<http://toniharyanto.cs.upi.edu>

# Variabel dan Operator

Haloo.. berjumpa lagi dengan NyanKod di pembahasan JavaScript. Kali ini kita akan belajar tentang variabel dan operator. Seperti yang sudah saya bahas di edisi kedua, tipe data di JavaScript pada dasarnya merupakan sebuah objek. Artinya ketika kita membuat sebuah variabel bertipe apapun, kita telah membuat sebuah objek. Dikatakan demikian karena variabel yang kita buat memiliki properties dan method yang dapat kita panggil dan gunakan sesuai dengan keperluan.

Properties dan method yang dimiliki sebuah variabel disesuaikan dengan tipe data atau dalam hal ini jenis objek yang dikandung oleh variabel tersebut. Contoh sederhana, ketika kita membuat sebuah variabel bertipe string, maka variabel tersebut memiliki beberapa method seperti `toUpperCase()` yang tidak dimiliki oleh variabel bertipe integer.

## Variabel

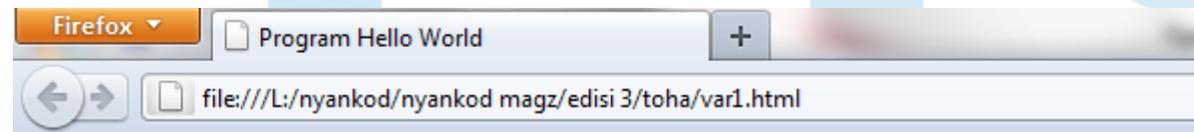
---

Untuk mengakomodir teman-teman yang baru belajar pemrograman, saya akan menjelaskan sedikit tentang apa itu variabel dan bagaimana cara membuatnya. Variabel adalah salah satu tokoh dalam pemrograman yang bertugas menyimpan sebuah nilai atau ekspresi. Nilai dalam sebuah variabel akan selalu ada selama program berjalan. Sebagai contoh, mari kita lihat script berikut.

```
1 | <html>
2 | <head><title>Program Hello World</title></head>
```

```
3 <body>
4 <script type="text/javascript">
5     var x;
6     x = "Hello World";
7     document.write(x);
8     var y = 10;
9     var z = 20;
10    var r = y + z;
11    document.write(r);
12    s = 25;
13    document.write(s);
14 </script>
15 </body>
16 </html>
```

Outputnya:



Hello World3025

Bila kita menjalankan program di atas (melalui browser tentunya) maka akan muncul tulisan "Hello World", angka 30 dan angka 25 pada browser. Beberapa penjelasan untuk contoh di atas berkaitan dengan variabel, diantaranya:

- Baris ke-5 adalah cara mendeklarasikan sebuah variabel, dalam program di atas variabel diberi nama x.
- Pada baris ke-6 variabel x diisi nilai "Hello World" yang bertipe string.

- Selain cara di atas, kita juga dapat mendeklarasikan sebuah variabel dan mengisinya dengan nilai sekaligus, seperti yang dicontohkan pada baris ke-8 dan ke-9.
- Selain dapat diisi oleh nilai, sebuah variabel juga dapat diisi oleh ekspresi, seperti dicontohkan pada baris ke-10. Pada baris tersebut variabel bernama `r` dideklarasikan dan diisi oleh ekspresi `y + z` artinya variabel `r` diisi oleh nilai dari variabel `y` ditambah nilai variabel `z`.
- Bila kita tidak mendeklarasikan sebuah variabel terlebih dahulu dan langsung mengisinya dengan nilai seperti pada baris ke-12, maka JavaScript akan otomatis mendeklarasikannya dan nilai yang kita berikan pada variabel tersebut akan tetap masuk.

Ada beberapa hal yang mesti kita perhatikan dalam membuat sebuah variabel, diantaranya:

- nama variabel adalah case-sensitive, artinya huruf besar dan huruf kecil dibedakan. Misalnya, variabel `y` dengan variabel `Y` adalah dua variabel yang berbeda.
- nama variabel hanya bisa diawali oleh huruf dan underscore. Beberapa nama variabel yang valid contohnya `x`, `y`, `jumlah2bilangan`, `Total`, `hasilPerhitungan` dan `_nilai_akhir`. Contoh nama variabel yang tidak valid seperti `123oye`, `#hahay` dan `!serius`.

Variabel ini akan sangat sering kita gunakan dalam pembuatan sebuah program. Selanjutnya kita akan belajar tentang operator.

## Operator

---

Operator adalah, emh, gimana yah penjelasannya? Kira-kira begini, operator adalah karakter atau simbol yang digunakan dalam sebuah program untuk melakukan pekerjaan tertentu. Pekerjaan tertentu yang dimaksud adalah operasi aritmatika,

operasi penugasan, operasi perbandingan, operator logika dan operasi konkatenansi. Pahami? Sepertinya Anda akan paham bila melihat contohnya secara langsung.

## Operator Penugasan

Pada contoh program di atas tadi, kita melihat contoh pengisian sebuah variabel oleh nilai tertentu. Nah, pengisian data atau nilai terhadap sebuah variabel disebut operasi penugasan, atau dalam bahasa Inggris dikenal dengan istilah Assignment Operation. Contoh,

```
1 | var x = "Hello World";
```

Pada contoh baris program di atas, tanda '=' yang digunakan untuk mengisi nilai "Hello World" pada variabel x adalah salah satu operator penugasan atau Assignment Operator. Beberapa operator penugasan lainnya dapat kita lihat di tabel berikut. Misalkan kita memiliki variabel  $x=5$  dan  $y=10$ , maka penerapan operator penugasan pada dua variabel tersebut adalah:

Operator	Contoh	Sama dengan	Hasil
=	$x=y$		$x=5$
+=	$x+=y$	$x=x+y$	$x=15$
-=	$x-=y$	$x=x-y$	$x=5$
*=	$x*=y$	$x=x*y$	$x=50$
/=	$x/=y$	$x=x/y$	$x=2$
%=	$x%=y$	$x=x\%y$	$x=0$

Pahami maksudnya? Penjelasannya seperti berikut.

- Operator '=' adalah operator yang paling umum kita gunakan dalam membuat program.  $x=y$  berarti nilai x diisi oleh nilai variabel y. Apabila x sudah memiliki nilai sebelumnya, maka nilai x lama akan tertimpa dan x akan diisi dengan nilai yang baru.

- Operator '+=' adalah operator untuk memperbaharui nilai variabel pertama dengan penjumlahan dari nilai variabel pertama dan nilai variabel kedua. pada contoh di atas, nilai x akan diperbaharui oleh nilai dari nilai variabel x yang lama ditambah nilai variabel y, atau dengan kata lain hasil dari operasi tersebut sama dengan operasi  $x = x + y$ .
- Operator '-=', '\*=', '/=' dan '%=' prosesnya sama dengan penjelasan kedua di atas tentang operator '+=', hanya karakter pertama saja yang membedakan.

## Operator Aritmatika

Operator aritmatika adalah operator yang sering kita gunakan dalam proses perhitungan. Anda pasti sudah tidak asing lagi dengan operator + untuk penjumlahan, - untuk pengurangan, \* untuk perkalian, / untuk pembagian, % untuk modulus (sisa pembagian), ++ untuk increment dan -- untuk decrement. Lengkapnya coba lihat tabel berikut ini.

Given that  $y=5$ , the table below explains the arithmetic operators:

Operator	Deskripsi	Contoh	Hasil
+	Penjumlahan	$x=y+2$	$x=7$
-	Pengurangan	$x=y-2$	$x=3$
*	Perkalian	$x=y*2$	$x=10$
/	Pembagian	$x=y/2$	$x=2.5$
%	Modulus	$x=y\%2$	$x=1$
++	Increment	$x=++y$	$x=6$
--	Decrement	$x=--y$	$x=4$

Mudah bukan??

## Operator Konkatenansi

Apabila kita memiliki dua buah variabel bertipe string, maka kita dapat menggabungkan kedua nilai variabel tersebut. Operasi ini dinamakan konkatenansi. Konkatenansi pada JavaScript menggunakan operator +. Kita dapat simpulkan bahwa, operator + berfungsi sebagai penjumlahan apabila kedua nilai yang mengapitnya berupa bilangan, dan berfungsi sebagai konkatenansi apabila salahsatu atau kedua nilai yang mengapitnya bertipe string. Langsung contohnya deh.

```
1 <html>
2 <head><title>Program Hello World</title></head>
3 <body>
4 <script type="text/javascript">
5     var x = "Haloo ";
6     var y = "Berapa umurmu?";
7     var z = x + y;
8     document.write(z);
9     var x = "Umurku ";
10    var y = 27;
11    var z = x + y;
12    document.write(z);
13 </script>
14 </body>
15 </html>
```

Pada contoh di atas, variabel z pada baris ke-7 bernilai "Haloo Berapa umurmu?" dan pada baris ke-11 variabel z bernilai "Umurku 27". Angka 27 pada variabel y di baris ke-10 menjadi string karena digabungkan dengan string dari variabel x.

## Operator Perbandingan

Operator perbandingan digunakan untuk membandingkan dua buah nilai atau variabel. Apabila kedua nilai yang dibandingkan sama, maka akan mengembalikan nilai true, dan bila tidak sama, maka akan mengembalikan nilai false.

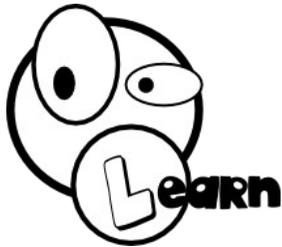
Misalkan kita punya variabel  $x=8$ , tabel di bawah ini akan menunjukkan contoh-contoh kasus untuk setiap operator.

Operator	Description	Example
==	sama dengan	$x==9$ bernilai false
===	tepat sama dengan (nilai dan tipe)	$x===8$ bernilai true $x==="8"$ bernilai false
!=	tidak sama dengan	$x!=9$ bernilai true
>	lebih dari	$x>9$ bernilai false
<	kurang dari	$x<9$ bernilai true
>=	lebih dari atau sama dengan	$x>=9$ bernilai false
<=	kurang dari atau sama dengan	$x<=9$ bernilai true

Operator perbandingan ini biasanya digunakan untuk menguji suatu nilai, dan pengujiannya dilakukan dengan menggunakan pernyataan kondisional atau conditional statement seperti if-else dan switch. Pembahasan itu nanti kita sajikan di edisi kedepannya.

## Operator Logika

Terakhir adalah operator logika. Operator logika yang dimaksud adalah and, or dan not, seperti yang pernah kita pelajari di pelajaran logika matematika. Kalo Anda lupa, silakan buka link berikut untuk sekedar penyegaran kembali [link]. Sama seperti



Sumber:

<http://www.W3school.com>

operator perbandingan, operator logika juga mengembalikan nilai true atau false atas suatu kondisi.

Misalkan kita punya variabel  $x=6$  dan  $y=3$ , maka contoh kasusnya dapat dilihat di tabel berikut.

Operator	Description	Example
&&	and	$(x < 10 \ \&\& \ y > 1)$ bernilai true
	or	$(x==5 \    \ y==5)$ bernilai false
!	not	$!(x==y)$ bernilai true

Untuk baris pertama, karena  $x < 10$  bernilai true dan  $y > 1$  bernilai true, maka true and true bernilai true. Untuk contoh kedua, karena  $x == 5$  bernilai false dan  $y == 5$  juga bernilai false, maka false or false bernilai false. Contoh terakhir, karena  $x == y$  bernilai true, maka not true bernilai false.

Demikian penjelasan sederhana tentang variabel dan operator. Sebenarnya masih ada satu jenis operator lagi, yaitu operator kondisional. Tapi berhubung operator ini ada kaitannya langsung dengan statemen if-else, maka kita tunda dulu penjelasannya di edisi berikutnya. ;D

## Cara Berlangganan Nyankod Magz via Email

Bagi Anda yang ingin berlangganan buletin Nyankod Magz via email, caranya gampang, udah gitu gratis lagi bro....

Gini nih caranya:

1. Buka homepage kami di <http://nyankod.blogspot.com>.
2. Pada sidebar sebelah kanan ada kolom Berlangganan Nyankod Magz. Silahkan isi form email dengan alamat email Anda, kemudian klik tombol Submit.

The screenshot shows a sidebar from the Nyankod Magz website. At the top, there is a blue banner with the text "learn coding here....". Below this is a search bar with the placeholder text "Search" and a magnifying glass icon. Underneath the search bar is a section titled "Berlangganan Nyankod Magz". This section contains a paragraph: "Dengan mengisi form di bawah ini, Anda akan mendapatkan email notifikasi setiap edisi terbaru Nyankod Magz." Below the paragraph is a form with an input field labeled "Email address..." and a "Submit" button. At the bottom of the sidebar, there is a "Recommend" button and a social media link for Facebook, which reads "Kresna Galuh BestFriend, Toni Y. Haryanto and 15".

3. Selanjutnya akan keluar pop-up Email Subscription Request. Isi form yang ada di bawah dengan kode yang tampil di layar. Setelah itu klik tombol Complete Subscription Request.



 FeedBurner™

## Email Subscription Request

Thank you for your request.

**nyankoders@email.com**

... will receive a verification message once you submit this form. FeedBurner activates your subscription to "Nyankod.com" once you respond to this verification message.



To help prevent spam, please type the text you see in the box above:

©2004–2012 Google ([Terms of Service](#) • [Privacy Policy](#))

4. Beres deh, nanti setiap ada edisi terbaru dari Nyankod Magz akan langsung terkirim pula ke email Anda. Biasanya sih satu hari setelah penerbitan.

Bagi Anda yang merasa Nyankod Magz bermanfaat, dan Anda ingin memberikan apresiasi, maka silahkan kirimkan Testimoni Anda kepada kami, atau silahkan tulis langsung di halaman Testimoni di web kami.

**Edisi ke-4 akan terbit: Rabu, 21 Maret 2012**



**Coding is pulen bro!!!**