PEMROGRAMAN BERORIENTASI OBJEK

Operator

Telkom University

Budhi Irawan, S.Si, M.T

OPERATOR

- **Operator** adalah tanda yang digunakan untuk melakukan operasi – operasi tertentu didalam program.
- Dengan adanya **operator** maka dapat melakukan ***operasi perhitungan, perbandingan, manipulasi bit*** dan lain-lain.
- Bahasa C++ merupakan salah satu bahasa pemrograman yang banyak menyediakan operator.

OPERATOR

Terdapat **empat kelompok operator** pada bahasa C++ diantaranya adalah :

- ***Operator Assigment,***
- ***Operator Unary,***
- ***Operator Binary,***
- ***Operator Ternary***.

OPERATOR

- Terdapat beberapa istilah yang harus diketahui dalam bekerja dengan operator, Contoh :

```
c = 5 + 7;
```

maka
c disebut ***variabel***
= disebut ***operator assignment***
5 dan 7 disebut ***operand***
5 + 7 disebut ***ekspresi***
+ disebut ***operator aritmatika (penambahan)***
c = 5 + 7 disebut ***statement aritmatika***

JENIS JENIS OPERATOR C++

Operator unary,
yaitu **Operator** yang digunakan dalam operasi yang hanya melibatkan ***satu buah operand*** contoh :

```
x++;
a = -b;
```

JENIS JENIS OPERATOR C++

Operator Binary, yaitu operator yang digunakan dalam operasi yang melibatkan ***dua buah operand*** contoh :

```
x = y + z;
a = 2 * 10;
```

JENIS JENIS OPERATOR C++

Operator Ternary, yaitu Operator yang digunakan dalam operasi yang melibatkan ***tiga buah operand*** contoh :

```
x = (x > 0) ? X : -x;
maks = (maks <= a} ?a : maks;
```

1. OPERATOR ASSIGNMENT

- ***Operator Assigenment (pengisian)*** adalah **operator** yang digunakan untuk memasukan atau mengisikan nilai kedalam suatu variabel.
- Dalam C++, operator yang digunakan untuk keperluan ini adalah **= (sama dengan)**. Contoh :

```
int a, b;
a = 72;
b = 54;
```

- Pada contoh diatas, akan dimasukan nilai `72` kedalam kedalam variabel `a` dan `54` kedalam variabel `b`.

1. OPERATOR ASSIGNMENT

- **Operator = (Sama Dengan)** dapat digunakan untuk mengisi nilai dari berbagai macam tipe data, bisa berupa bilangan (bulat dan riil), karakter, boolean, string, maupun tipe data bentukan lainnya.

I. OPERATOR ASSIGNMENT

```
#include <iostream>
using namespace std;
int main()
{
  // deklarasi variabel
  int i;
  double d;
  char c;
  char *s;
  // mengisi nilai ke dalam variabel
  i = 75;
  d = 8.615;
  c = 'C';
  s = (char *) "Contoh string";
  // menampilkan nilai variabel
  cout<<"Nilai i : "<<i<<endl;
  cout<<"Nilai d : "<<d<<endl;
  cout<<"Nilai c : "<<c<<endl;
  cout<<"Nilai s : "<<s<<endl;

  return 0;
}
```

I. OPERATOR ASSIGNMENT

- Dalam kode program sering ditemui statement berikut :

```
i = i + 1;
```

statement tersebut memiliki arti :

$i_{baru} = i_{lama} + 1;$

- Dalam C++, statement seperti diatas dapat disingkat penulisannya menjadi :

```
i += 1;
```

1. OPERATOR ASSIGNMENT

- Bentuk singkat penulisan operator = (sama dengan) juga berlaku untuk operasi lainnya seperti : pengurangan, perkalian, pembagian, perhitungan sisa bagi dan sebagainya. Contoh :

```
a += b; // sama dengan a = a + b;
a -= b; // sama dengan a = a - b;
a *= b; // sama dengan a = a * b;
a /= b; // sama dengan a = a / b;
a %= b; // sama dengan a = a % b;
```

2. OPERATOR UNARY

- Dalam ilmu matematika yang disebut dengan ***Operator Unary*** adalah **Operator** yang hanya melibatkan sebuah ***operand***.
- Beberapa operator yang termasuk kedalam ***Operator unary*** adalah :

Operator	Jenis Operasi	Contoh
+	Membuat nilai positif	+7
-	Membuat nilai negatif	-7
++	Increment	C++
--	Decrement	C--

2. OPERATOR UNARY

```
#include <iostream>
using namespace std;

int main() {
  int X;
  float Y;

  X = +15;  // Dapat ditulis dengan X = 15,
          // yang berarti memasukkan nilai positif 15
  Y = -4.32;  // Memasukkan nilai negatif 4.32

  // Menampilkan nilai
  // yang disimpan dalam variabel X dan Y
  cout<<"Nilai X : "<<X<<endl;
  cout<<"Nilai Y : "<<Y<<endl;

  X = -X; // Mengubah nilai X menjadi negatif
  Y = -Y;

  // Menampilkan kembali nilai yang disimpan
  // dalam variabel X dan Y
  cout<<"Nilai X : "<<X<<endl;
  cout<<"Nilai Y : "<<Y<<endl;

  return 0;
}
```

INCREMENT

- ***Increment*** adalah suatu penambahan nilai yang terjadi pada sebuah variabel.
- Adapun operator yang digunakan untuk melakukan ***increment*** adalah **operator ++**
- Operator ini akan menambahkan nilai dari suatu variabel dengan nilai 1

INCREMENT

- Terdapat dua jenis increment yang terdapat dalam bahasa C++ yaitu ***pre-increment*** dan ***post-increment***.
- ***Pre-Increment*** adalah melakukan penambahan nilai sebelum suatu variabel itu diproses, sedangkan ***Post-Increment*** merupakan kebalikannya, yaitu melakukan proses terlebih dahulu sebelum dilakukan penambahan nilai.

INCREMENT

Bentuk umum dari ***pre-increment*** dan ***post-increment*** adalah :

// Melakukan ***pre-increment***

```
++nama_variabel;
```

// Melakukan ***post-increment***

```
nama_variabel++;
```

INCREMENT

```
#include <iostream>

using namespace std;

int main() {

  int C; // Mendeklarsikan variabel C

  // Mengisikan nilai ke dalam variabel C
  // dengan nilai 15
  C = 15;

  // Melakukan pre-increment
  cout<<"Nilai C awal : "<<C<<endl;    cout<<"Nilai ++C  : "<<++C<<endl;
  cout<<"Nilai C akhir : "<<C<<endl;  cout<<'\n';

  // Mengubah nilai yang terdapat dalam variabel C
  // dengan nilai 30
  C = 30;

  // Melakukan post-increment
  cout<<"Nilai C awal : "<<C<<endl;  cout<<"Nilai C++ : "<<C++<<endl;
  cout<<"Nilai C akhir : "<<C<<endl;

  return 0;
}
```

DECREMENT

- ***Decrement*** merupakan kebalikan dari proses ***increment*** , yaitu menurunkan (mengurangi) nilai dari suatu variabel.
- Sama juga seperti pada ***increment*** , ***decrement*** juga dibagi kedalam dua jenis yaitu ***pre-decrement*** dan ***post-decrement***

3. OPERATOR BINARY

- Operator ***Binary*** adalah operator yang digunakan dalam operasi yang melibatkan 2 buah ***operand***
- Dalam bahasa C++ operator binary ini dikelompokan lagi kedalam 4 jenis yaitu ***a. Operator aritmetika***, ***b. Operator logika***, ***c. Operator relasional*** dan ***d. Operator bitwise***.

3A. OPERATOR ARITMETIKA

- ***Operator Aritmetika*** adalah operator yang digunakan untuk melakukan operasi-operasi aritmetika seperti penjumlahan, pengurangan dan sebagainya

Operator	Jenis Operasi	Contoh
+	Penjumlahan	2 + 3 = 5
-	Pengurangan	5 – 3 = 2
*	Perkalian	2 * 3 = 6
/	Pembagian	10.0 / 3.0 = 0.33333
%	Sisa Hasil Bagi (Modulus)	10% 3 = 1

3A. OPERATOR ARITMETIKA (+)

```
#include <iostream>

using namespace std;

int main() {
  // Mendeklarasikan variabel X (diisi nilai 10)
  // dan Y (diisi nilai 3)
  int X = 25, Y = 15;
  // Mendeklarasikan variabel Z sebagai penampung
  // nilai hasil operasi
  int Z;

  // Melakukan operasi penjumlahan
  Z = X + Y;

  // Menampilkan hasil penjumahan
  cout<<X<<" + "<<Y<<" = "<<Z;

  return 0;
}
```

3B. OPERATOR LOGIKA

- Operator ***Logika*** adalah operator yang digunakan untuk melakukan operasi dimana nilai yang dihasilkan dari operasi tersebut hanya berupa nilai benar ***(true)*** dan salah ***(false)***
- Nilai ini disebut nilai ***boolean***

3B. OPERATOR LOGIKA

- Dalam bahasa C++, nilai benar tersebut di representasikan dengan nilangan selain 0 (biasanya nilai 1), sedangkan nilai salah direperensasikan dengan nilai 0 .
- Namun dalam C++ modern yang telah mendukung tipe **`bool`** , nilai benar direperentasikan dengan nilai **`true`** dan nilai salah dengan nilai **`false`**

3B. OPERATOR LOGIKA

Operator	Jenis Operasi	Contoh
`&&`	AND (dan)	`1 && 1 = 1`
`||`	OR (atau)	`1 || 0 = 1`
`!`	NOT (negasi)	`!0 = 1`

OPERATOR LOGIKA && (AND)

- Operator ***AND*** hanya akan menghasilkan nilai 1 (benar) jika semua operand-nya bernilai benar. Namun jika tidak maka operasi tersebut akan menghasilkan nilai 0 (salah)

X	Y	X && Y
1	1	1
1	0	0
0	0	0
0	1	0

OPERATOR LOGIKA && (AND)

```
#include <iostream>

using namespace std;

int main() {
   cout<<"1 && 1 = "<<(1 && 1)<<endl;
   cout<<"1 && 0 = "<<(1 && 0)<<endl;
   cout<<"0 && 0 = "<<(0 && 0)<<endl;
   cout<<"0 && 1 = "<<(0 && 1)<<endl;

   return 0;
}
```

OPERATOR LOGIKA | | (OR)

- Operator ***OR*** hanya akan menghasilkan nilai 0 (salah) jika semua operand nya bernilai salah, namun jika tidak maka operasi tersebut akan menghasilkan nilai 1 (benar)

X	Y	X && Y
1	1	1
1	0	1
0	0	0
0	1	1

OPERATOR LOGIKA ! (NOT)

- Operator ***OR*** hanya akan menghasilkan nilai kebalikan dari nilai yang dikandung didalamnya. Jika nilai awalnya adalah **1** (benar), maka setelah operasi **NOT** nilanya akan menjasi 0 (salah) begitu juga sebaliknya

x	!x
1	0
0	1

3C. OPERATOR RELATIONAL

- Operator ***Relasional*** adalah operator yang digunakan untuk menentukan relasi atau hubungan dari dua ***operand***.
- Operator ini ditempatkan didalam sebuah ekspresi yang kemudian akan menentukan benar atau tidaknya sebuah ekspresi.

3C. OPERATOR RELATIONAL

Operator	Jenis Operasi	Contoh
>	Lebih Besar	(5 > 2) = 1
<	Lebih Kecil	(5 < 2) = 0
>=	Lebih Besar atau Sama Dengan	(5 >= 5) = 1
<=	Lebih Kecil atau Sama Dengan	(5 <= 2) = 0
==	Sama Dengan	(5 == 2) = 0
!=	Tidak Sama Dengan	(5 != 2) = 1

3D. OPERATOR BITWISE

- ***Operator Bitwise*** berguna untuk melakukan operasi – operasi yang berhubungan dengan manipulasi bit sejak bahasa C diciptakan,
- Bahasa C banyak digunakan untuk lebih memudahkan pemrograman yang berhubungan dengan alat karena bahasa tersebut mendukung operasi-operasi bitwise yang biasanya dilakukan dengan menggunakan bahasa assembly.

3D. OPERTOR BITWISE

- Operator Bitwise hanya dapat dilakukan pada operand yang bertipe **`char`** dan **`int`** saja karena ini berkoresponden dengan type ***byte*** atau ***word*** didalam bit.

Operator	Jenis Operasi	Contoh
&	AND	`1 & 0 = 0`
|	OR	`1 | 0 = 1`
^	Exclusive OR (XOR)	`1 ^ 1 = 0`
~	NOT	`~1 = 0`
>>	SHIFT RIGHT	`5 << 1 = 10`
<<	SHIFT LEFT	`10 >> 1 = 5`

OPERATOR ^

- **Operator ^** berguna untuk melakukan ***operasi eXclusive OR*** (XOR).
- Adapun hasil dari operasi ini akan benilai 1 (benar) jika satu operand nya (bukan salah satu) bernilai benar, selain itu akan menghasilkan nilai 0 (salah).
- Sehingga jika kedua operand nya bernilai 1 (benar), maka hasil dari operasi ini adalah 0 (salah).

OPERATOR ^

X	Y	X ^Y
1	1	0
1	0	1
0	0	0
0	1	1

OPERATOR >>

- ***Operator Shift Right*** **>>** berguna untuk melakukan perpindahan bit ke arah kanan. Adapun bentuk dari penggunaan operator ini adalah :

```
nilai>>banyaknya_pergeseran_bit_kearah_kanan
```

OPERATOR >>

```
#include <iostream>

using namespace std;

int main() {
  int X, Y;

  // Menggeser 1 bit ke kanan
  // dari bentuk biner bilangan 16
  X = 16 >> 1;

  // Menggeser 2 bit ke kanan
  // dari bentuk biner bilangan 16
  Y = 16 >> 2;

  // Menampilkan hasil
  cout<<"16 >> 1 = "<<X<<endl;
  cout<<"16 >> 2 = "<<Y;

  return 0;
}
```

OPERATOR >>

Ilustrasi Proses :

Nilai X	X dalam Biner	Hasil
x = 16	00010000	16
x = 16 >> 1	00001000	8
x = 16 >> 2	00000100	4
x = 16 >> 3	00000010	2
x = 16 >> 4	00000001	1

OPERATOR <<

- **Operator Shift Left <<** berguna untuk melakukan perpindahan bit ke arah kiri. Adapun bentuk dari penggunaan operator ini adalah :

```
nilai<<banyaknya_pergeseran_bit_kearah_kiri
```

OPERATOR >>

Ilustrasi Proses :

Nilai X	X dalam Biner	Hasil
`x`	00000001	1
`x = 1 << 1`	00000010	2
`x = 1 << 2`	00000100	4
`x = 1 << 3`	00001000	8
`x = 1 << 4`	00010000	16

4. OPERATOR TERNARY

- Operator **Ternary** adalah operator yang digunakan dalam operasi yang melibatkan tiga buah operand.
- Adapun operator yang digunakan untuk menyatakannya adalah operator **? :**
- Konsep yang mendasari operasi ini adalah suatu percabangan (pemilihan) yang didasarkan atas kondisi tertentu.

4. OPERTOR TERNARY

```
Ekspresi1? Ekspresi2? Ekspresi3;
```

- Jika **`Ekspresi1`** bernilai benar, maka program akan mengeksekusi **`Ekspresi2`**.
- Sedangkan jika **`Ekspresi1`** bernilai salah maka yang dieksekusi adalah **`Ekspresi3`**

5. OPERTOR TERNARY

```
#include <iostream>

using namespace std;

int main() {

  int X;

  // Meminta user untuk memasukkan nilai X
  // dari keyboard
  cout<<"Masukkan nilai X : "; cin>>X;
  cout<<'\n';

  // Melakukan pemeriksaan terhadap nilai X
  X = (X < 0) ? -X : X;

  // Menampilkan nilai X
  // setelah proses pemeriksaan
  cout<<"| X | = "<<X;

  return 0;
}
```