OPERATOR DAN UNGKAPAN

5

Overview

Operator merupakan simbol yang biasa dilibatkan dalam program untuk melakukan suatu operasi atau manipulasi, sedangkan ungkapan adalah dasar bagi pernyataan berkondisi (misalnya **if**). Hasil ungkapan berupa 1 kalau ungkapan bernilai benar dan ungkapan berupa 0 kalau ungkapan bernilai salah.

Tujuan

- ☑ Mengenal dan membedakan operator dan ungkapan
- ☑ Mengetahui jenis-jenis operator dan ungkapan
- ☑ Memahami operator dan penggunaannya dalam program
- ☑ Memahami ungkapan dan penggunaannya dalam program

5.1 Pengantar Operator dan Ungkapan

Operator merupakan simbol yang biasa dilibatkan dalam program untuk melakukan suatu operasi atau manipulasi. Sebagaian operator C++ tergolong sebagai operator *binary*, yaitu operator yang dikenakan terhadap dua buah nilai (*operand*).

Contoh :

a + b

simbol " + " merupakan *operand* untuk melakukan penjumlahan dari a dan b. Karena operator penjumlahan melibatkan dua *operand*, operator penjumlahan tergolong sebagai operator *binary*.

Contoh lain :

-c

simbol " – " (minus) merupakan *unary*, karena hanya memiliki sebuah *operand* (yaitu c pada contoh diatas).

Ungkapan (ekspresi) dalam C++ dapat berupa :

a. Pengenal
b. Konstanta
c. Diantara kombinasi elemen diatas dengan operator

Contoh ungkapan :

3 + 2 - 1

Pada ungkapan diatas, 3, 2 dan 1 merupakan operand dan simbol "+" serta "–" adalah operator. Nilai ungkapan sendiri adalah hasil penjumlahan 3 dan 2, dikurangi 1.

5.1.1 Operator Aritmatika

Operator untuk aritmatika yang tergolong sebagai operator *binary*. Contoh penggunan operator aritmatika misalnya untuk memperoleh nilai diskriminan dari suatu persamaan kuadrat.

$d = b^2 - 4ac$

untuk mengimplementasikan contoh diatas adalah seperti berikut:

```
d = b * b – 4 * a * c ;
```

Contoh 5	Hasil

//*----------*
//* contoh 5 : Contoh pemakaian operator *
//* Aritmatika *
//*----------*
#include <iostream.h>
#include <conio.h>
int main()
{
int a, b, c, d;
clrscr();
a = 5;
b = 600;
c = 5;
 	
d = b * b – 4 * a * c;
cout << " d = " << d << '\n';
return 0;
}
	d = 32220

Operator aritmatika mempunyai prioritas pengerjaan. Prioritas yang tinggi akan diuatamakan dalam hal pengerjaan dibandingkan dengan operator yang memiliki prioritas yang lebih rendah. Urutan prioritas dapat dilihat dalam tabel berikut ini :

Operator	Prioritas
+ -- (Khusus yang berkedudukan sebagai awalan)	Tertinggi
- (Unary Minus)	
* / %	
+ -	Terendah

Jika operator memiliki prioritas yang sama, operator sebelah kiri akan diutamakan untuk dikerjakan terlebih dahulu. Tanda kurung biasa digunakan untuk merubah urutan pengerjaan.

Misalnya : x = (2 + 3) * 2 ;

akan memberikan nilai 10 ke x, sebab 2 + 3 dikerjakan terlebih dahulu dan hasilnya baru dikalikan dengan 2.

Contoh 6	Hasil
//*----------* //* Contoh 6 : Penggunaan kurung untuk mengatur * //* prioritas pengerjaan terhadap suatu * //* operasi * //*----------* #include <iostream.h> #include <conio.h> void main() { clrscr()	X = 8 X = 12

```
 int x ;
 x = 2 + 3 * 2 ;
 cout << " x = " << x << '\n';
 x = (2 + 3) * 2 ;
 cout << " x = " << x << '\n';
return 0;
}
```

5.1.2 Operator Sisa Pembagian

Operator sisa pembagian (operator modulus) yang berupa %. Operator ini diterapkan pada operand bertipe integer. Untuk lebih jelasnya perhatikan contoh berikut :

7 % 2 → 1	Sisa pembagian bilangan 7 dengan 2 adalah 1
6 % 2 → 0	Sisa pembagian bilangan 6 dengan 2 adalah 0
8 % 3 → 2	Sisa pembagian bilangan 8 dengan 3 adalah 2

Contoh 7

```
//*---------------------------------------------------*
//* Contoh 7 : Melihat sisi pembagian dengan *
//* menggunakan operator %. *
//*---------------------------------------------------  *
#include <iostream.h>
#include <conio.h>

void main()
{
 clrscr();
 cout << 5 % 7 << '\n'; //sisa 5
 cout << 6 % 7 << '\n'; //sisa 6

 cout << 7 % 7 << '\n'; //sisa 0
 cout << 8 % 7 << '\n'; //sisa 1
 cout << 9 % 7 << '\n'; //sisa 2
return 0;
}
```

Hasil

```
5
6
0
1
2
```

Kegunaan operator % diantaranya bisa dipakai untuk menentukan suatu bilangan bulat termasuk ganjil atau genap.

5.1.3 Operator Penurunan dan Penaikan

Kedua operator ini digunakan pada operand bertipe bilangan bulat. Operator penaikan digunakan untuk menaikan nilai variabel

sebesar satu, sedangkan operator penurunan dipakai untuk menurunkan nilai variabel sebesar satu. Sebagai contoh :

```
x = x + 1 ;
y = y – 1 ;
```

bisa ditulis menjadi :

```
++ x ;
-- y ;
```

atau :

```
x ++ ;
y -- ;
```

5.1.3.1 Penaikan dibelakang

Efek peletakkan tanda ++ dibelakang variabel ditunjukkan pada program berikut :

Contoh 8	Hasil

//*----------*
//* Contoh 8 : Pemakaian operator penaikan di *
//* belakang variabel *
//*----------*
#include <iostream.h>
#include <conio.h>
void main()
{
int r = 10;
int s;
clrscr();
s = 10 + r++ ;
cout << "R = " << r << '\n' ;
 	
cout << "S = " << s << '\n' ;
return 0;
}
	R = 11
	S = 20

Pada contoh diatas s diisi dengan penjumlahan nilai 10 dan r. Dengan demikian s akan bernilai 20. setelah s diisi dengan 20, nilai r baru dinaikan karena operator ++ ditulis dibelakang r. Disebut *post-increment* yang artinya dinaikkan dibelakang setelah penjumlahan anatara r dan 10 dilaksanakan.

5.1.3.2 Penaikan di Depan

Efek peletakkan tanda ++ di depan variabel ditunjukkan pada program berikut ini :

Contoh 9	Hasil

//*----------*
//* Contoh 9: Pemakaian operator penaikan di *
//* belakang variabel *
//*----------*
 	
#include <iostream.h>
#include <conio.h>
void main()
{
int r = 10;
int s;
clrscr();
s = 10 + ++r ;
cout << "R = " << r << '\n' ;
cout << "S = " << s << '\n' ;
return 0;
}
	R = 11
	S = 21

Pada contoh ini, nilai r mula-mula dinaikan terlebih dahulu karena operator ++ ditempatkan didepan r. Disebut *pre-increment* kemudian nilainnya dijumlahkan dengan 10 dan diberikan ke s. Dengan demikian s bernilai 21 dan r sama dengan 11.

5.1.4 Operator Majemuk

Operator majemuk digunakan untuk memendekkan penulisan operasi penugasan semacam :

```
x = x + 2 ;
y = y * 4 ;
```

menjadi :

```
x += 2;
y *= 4;
```

Contoh 10	Hasil

//*----------*
//* Contoh 10 : penggunaan operator majemuk *
//*----------*
#include <iostream.h>
#include <conio.h>
void main()
{
int x = 2; // Mula-mula x bernilai 2
clrscr();
cout << "x = " << x << '\n' ;
x += 3 ;
cout << "Setelah x += 3, x = " << x << '\n' ;
x *= 2 ;
cout << "Setelah x *= 2, x = " << x << '\n' ;
return 0;
}
	x = 2
	Setelah x += 3, x = 5
	Setelah x += 3, x = 5

5.1.5 Operator Kondisi

Operator kondisi biasa dipakai untuk mendapatkan sebuah nilai dari dua buah kemungkinan, berdasarkan suatu kondisi. Format pemakaiannya :

ungkapan1 ? ungkapan 2 : ungkapan 3

Contoh 11	Hasil

//*----------*
//* Contoh 11 : Penggunaan operator kondisi untuk *
//* memperoleh bilangan terkecil *
//* diantara dua buah bilangan *
//*----------*
#include <iostream.h>
#include <conio.h>
void main()
{
int bil1, bil2, minim;
clrscr();
bil1 = 53;
bil2 = 6;
minim = bil1 < bil2 ? bil1 : bil2;
cout << " Bilangan terkecil = " << minim << '\n';
 	
return 0;
}
	Bilangan terkecil =6

minim = bil1 < bil2 ? bil1 : bil2;

akan menyebabkan minim bernilai bil1 kalau ungkapan :

bil1 < bil2
bernilai benar. Untuk keadaan sebaliknya, minim akan bernilai bil2.

5.2 Ungkapan Kondisi

Ungkapan adalah ungkapan yang menjadi dasar bagi pernyataan berkondisi (misalnya **if**). Hasil ungkapan berupa 1 kalau ungkapan bernilai benar dan ungkapan berupa 0 kalau ungkapan bernilai salah.

5.2.1 Operator Relasi

Operator biasa digunakan untuk membandingkan dua buah nilai. Macam operator relasi dapat dilihat dalam tabel berikut :

Operator	Keterangan
==	Sama dengan (bukan penugasan)
!=	Tidak sama dengan
>	Lebih dari
<	Kurang dari
>=	Lebih dari atau sama dengan
<=	Kurang dari atau sama dengan

Agar tidak salah dalam menuliskan suatu ungkapan, pengetahuan tentang prioritas operator perlu diketahui.

Contoh 1 :

a = b = c

pada pernyataan diatas, operator yang dilibatkan (=) mempunyai sifat pengerjaan dimulai dari kanan. Berarti :

b = c

akan dikerjakan terlebih dahulu, barulah kemudian mengerjakan :

a = b

Contoh 2 : x = 2 * 3 * 4 ;

pada pernyataan diatas, 2 * 3 akan dikerjakan terlebih dahulu, barulah kemudian mengerjakan perkalian hasil 6 dengan 4. Adapun prioritas = lebih rendah dari *, maka 2 * 3 * 4 dikerjakan lebih dahulu. Selanjutnya hasilnya baru diberikan ke x.

5.2.2 Fungsi Pustaka

Dalam melakukan operasi seperti memeperoleh akar kuadrat ataupun memeperoleh logaritma alamiah dari suatu nilai. Pada C++ memang tidak terdapat operator-operator yang khusus untuk melaksanakan operasi-oerasi seperti itu. Tetapi tidak berarti C++ tidak dapat melakukan operasi itu. C++ menyediakan sejumlah fungsi pustaka (*library fuctions*) yang dirancang untuk memenuhi solusi dari berbagai persoalan.

Misalkan kita akan menghitung sebuah akar kuadrat, pemrogram bisa menggunakan fungsi **sqrt()**.

Jika program ingin menggunakan fungsi pustaka, perlulah untuk mencatumkan deklarasi dari fungsi bersangkutan. Untuk keperluan ini program mesti menyertakan baris :

```
#include <nama_file>
```

dengan *nama_file* adalah nama *header*, yaitu file yang berakhiran **.h**. sebagai contoh program diatas menyertakan **#include <math.h>** disebabkan file *header* tersebut berisi deklarasi (prototipe) dari fungsi **sqrt()**.

Rangkuman

- ☑ Tipe data dasar adalah tipe data yang dapat langsung digunakan, dan memiliki ukuran tertentu sesuai dengan tipe data masing-masing. Beberapa tipe data dasar: integer, real, char, string, dan boolean.
- ☑ Variabel adalah obyek yang nilainya dapat berubah-ubah dalam sebuah program, sedangkan konstanta memiliki nilai yang tetap sejak dideklarasikan hingga program berakhir.
- ☑ Pada saat mendeklarasikan sebuah variabel, pemrogram harus menyebutkan nama variabel dan tipe data dari variabel tersebut.
- ☑ Operator adalah pengendali operasi yang akan dilakukan pada beberapa operan sehingga membentuk sebuah ekspresi. Tiga macam operator dalam pemrograman: operator aritmatik, operator relasional, dan operator logika.
- ☑ Penggunaan beberapa operator sekaligus dalam sebuah ekspresi mengakibatkan perlunya menentukan urutan operasi yang diatur dalam level urutan operator.
- ☑ Operator dengan level yang lebih tinggi akan dieksekusi lebih dahulu daripada operan dengan level yang lebih rendah. Operator-operator yang memiliki level yang sama akan dieksekusi menurut urutan penulisan (dari kiri ke kanan).