# BAB II

# OPERATOR DAN STATEMENT I/O

## Operator

Operator merupakan simbol yang biasa dilibatkan dalam program untuk melakukan suatu operasi atau manipulasi. Dengan operator, kita dapat melakukan operasi perhitungan, perbandingan, manipulasi bit, dan lain-lain.

Sebagai pengenalan, terdapat istilah yang harus diketahui dalam bekerja dengan operator. Sebagai contoh :

|  |
| --- |
| A = 4 + 12; |

Keterangan :

* A disebut variable
* = disebut operator assignment
* 4 dan 12 disebut operand
* 4 + 12 disebut ekspresi
* + disebut operator aritmatika (penambahan)
* A = 4 + 12 disebut statemen aritmatika

Pada dasarnya, ada tiga jenis operator dalam pemrograman C++

* *Unary*

Sifat *unary* pada operator adalah melibatkan sebuah *operand* pada suatu operasi aritmatika

Contoh: -5

* *Binary*

Sifat *binary* pada operator adalah melibatkan 2 buah *operand* pada suatu operasi artimatika.

Contoh: 4 + 8

* *Ternary*

Sifat *ternary* pada operator adalah melibatkan 3 buah *operand* pada suatu operasi aritmatika

Contoh: 10 % 3 + 2;

a > b ? 1 : 0;

### Operator Assingment

Operator assignment (penugasan) adalah operator yang digunakan untuk memasukkan atau mengisikan nilai ke dalam suatu variabel. Dalam bahasa C++ digunakan operator berupa tanda sama dengan ( “ = ” ).

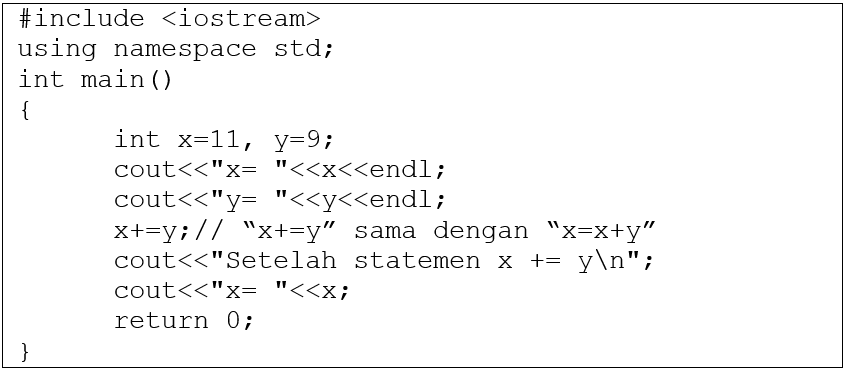
Contoh :

|  |
| --- |
| int A, B;  A = 55;  B = 66; |

Pada contoh di atas, kita memasukkan nilai 55 ke dalam variabel A dan 66 ke dalam variabel B.

**Latihan !**

1. Buatlah hasil output program berikut dan jelaskan!



### Operator Aritmatika

Operator ini digunakan untuk melakukan sebuah opearasi penghitungan atau aritmatika. Tugasnya untuk memecahkan masalah matematika.

Tabel operator aritmatika :

|  |  |  |
| --- | --- | --- |
| Operator | Deskripsi | Contoh |
| + | Penjumlahan | X + Y |
| - | Pengurangan | X - Y |
| \* | Perkalian | X \* Y |
| / | Pembagian | X / Y |
| % | Sisa pembagian Integer (Modulus) | X % Y |

Pada operator penjumlahan, pengurangan, perkalian, dan pembagian dilakukan operasi seperti biasa. Sedangkan, yang di maksud dengan sisa pembagian adalah **sisa dari hasil pembagian** bukan hasil dari pembagian.

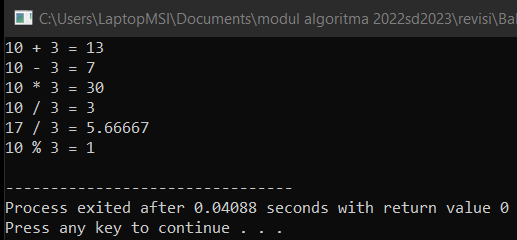
Contohnya :

|  |
| --- |
| 10 % 2  Hasil pembagian = 5  Sisa pembagian = 0 |

Contoh :

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  int i=10 , j=3;  float k=17.0 , l=3.0;  cout<<i<<" + "<<j<<" = "<<i+j<<endl;  cout<<i<<" - "<<j<<" = "<<i-j<<endl;  cout<<i<<" \* "<<j<<" = "<<i\*j<<endl;  cout<<i<<" / "<<j<<" = "<<i/j<<endl;  cout<<k<<" / "<<l<<" = "<<k/l<<endl;  cout<<i<<" % "<<j<<" = "<<i%j<<endl;  return 0;  } |

Tampilan :



Analisa :

Pada operator pembagian, pembagian yang pertama menggunakan tipe data int maka hasilnya jadi 3, sedangkan yang kedua menggunakan tipe data float maka hasilnya akan pecahan.

Pada sebuah ekspresi bisa saja dijumpai lebih dari 1 operator. Contohnya: x = 2 + 6 / 3; Dalam C++ variable x menghasilkan nilai 4, karena operator “ / ” memiliki kedudukan yang lebih tinggi dari operator “ + ”.

Dibawah ini adalah tabel prioritas operator:

|  |  |  |
| --- | --- | --- |
| Kedudukan | Operator | Deskripsi |
| Tertinggi | ++ -- | Increment, dercement |
| \* / % | Perkalian, pembagian, modulus |
| Terendah | + - | Penjumlahan, pengurangan |

### Operator Relasional

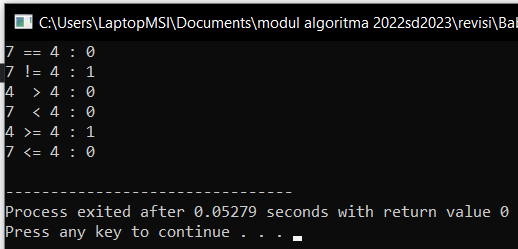
Operator relasional atau hubungan digunakan untuk membandingkan hubungan antara dua buah operand. Pada operator ini menghasilkan kondisi **benar** atau **salah.**

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Deskripsi | Contoh | |
| == | Sama dengan  (bukan assigment) | 5==5 | Benar, karena 5 sama dengan 5 |
| != | Tidak sama dengan | 5!=2 | Benar, karena 5 tidak sama dengan 2 |
| > | Lebih besar | 5 > 2 | Benar, karena 5 lebih besar dari 2 |
| < | Lebih Kecil | 5 < 2 | Tidak, karena 5 lebih besar dari 2 |
| >= | Lebih besar atau sama dengan | 5>=5 | Benar, karena 5 sama dengan 5 |
| <= | Lebih kecil atau sama dengan | 5<=2 | Tidak, karena 5 lebih besar dari 2 |

Contoh :

|  |
| --- |
| #include <iostream>  using namespace std;  main(){  int a,b,c,d,e,f;  int x=7, y=4, z=4;  a = x==y;  b = x!=y;  c = y>z;  d = x<y;  e = y>=z;  f = x<=y;  cout << x << " == " << y << " : " << a <<endl;  cout << x << " != " << y << " : " << b <<endl;  cout << y << " > " << z << " : " << c <<endl;  cout << x << " < " << y << " : " << d <<endl;  cout << y << " >= " << z << " : " << e <<endl;  cout << x << " <= " << y << " : " << f <<endl;  } |

Tampilan :



Analisa :

Pada program diatas dilakukan operasi menggunakan operator relasional. Bernilai 0 jika salah (false) dan 1 jika benar (true).

### Operator Naik dan Turun (Increment dan Decrement)

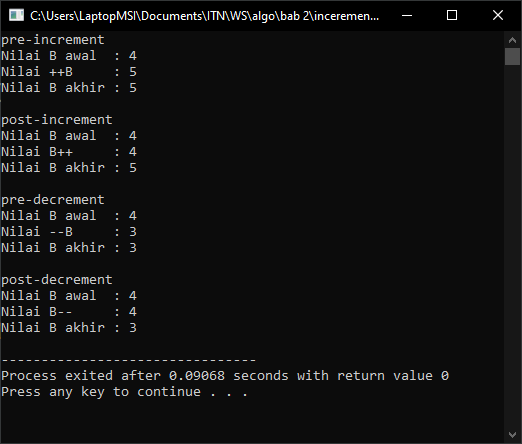
Operator *increment* merupakan operator yang dapat menambahkan (menaikkan) suatu nilai. Operator i*ncrement* ini ditandai dengan tanda **“++”.** Sedangkan, operator *decrement* merupakan operator yang mengurangkan (menurunkan) suatu nilai. Operator *decrement* ini ditandai dengan tanda **“ -- ”.**

Terdapat dua jenis increment/decrement, yaitu pre-increment adalah melakukan penambahan nilai sebelum suatu variabel itu diproses, dan post-increment yaitu melakukan proses terlebih dahulu sebelum dilakukan penambahan nilai.

Contoh :

|  |
| --- |
| #include <iostream>  using namespace std;  int main(){  int B;  B = 4;  cout<<"pre-increment"<<endl;  cout<<"Nilai B awal : "<<B<<endl;  cout<<"Nilai ++B : "<<++B<<endl;  cout<<"Nilai B akhir : "<<B<<endl;  B = 4;  cout<<"\npost-increment"<<endl;  cout<<"Nilai B awal : "<<B<<endl;  cout<<"Nilai B++ : "<<B++<<endl;  cout<<"Nilai B akhir : "<<B<<endl;  B = 4;  cout<<"\npre-decrement"<<endl;  cout<<"Nilai B awal : "<<B<<endl;  cout<<"Nilai --B : "<<--B<<endl;  cout<<"Nilai B akhir : "<<B<<endl;  B = 4;  cout<<"\npost-decrement"<<endl;  cout<<"Nilai B awal : "<<B<<endl;  cout<<"Nilai B-- : "<<B--<<endl;  cout<<"Nilai B akhir : "<<B<<endl;  return 0;  } |

Tampilan :



Analisa :

Pada saat kita melakukan pre-increment, nilai D dinaikan dulu sebelum diproses sehingga yang tampil adalah nilai 6. Sedangkan saat melakukan post-increment nilai dari variabel D diproses lebih dahulu sebelum dinaikan nilainya. Hal ini menyebabkan nilai yang ditampilkan masih tetap 6. Namun pada saat pemanggilan nilai D setelah proses post-increment, nilai D tentu telah bertambah satu, sehingga yang ditampilkan adalah 7, begitu juga sebaliknya untuk decrement.

### Operator Majemuk (Compound Assignment)

Operator majemuk digunakan untuk memendekkan penulisan operasi penugasan semacam :

|  |
| --- |
| x = x + 2 ;  y = y \* 4 ; |

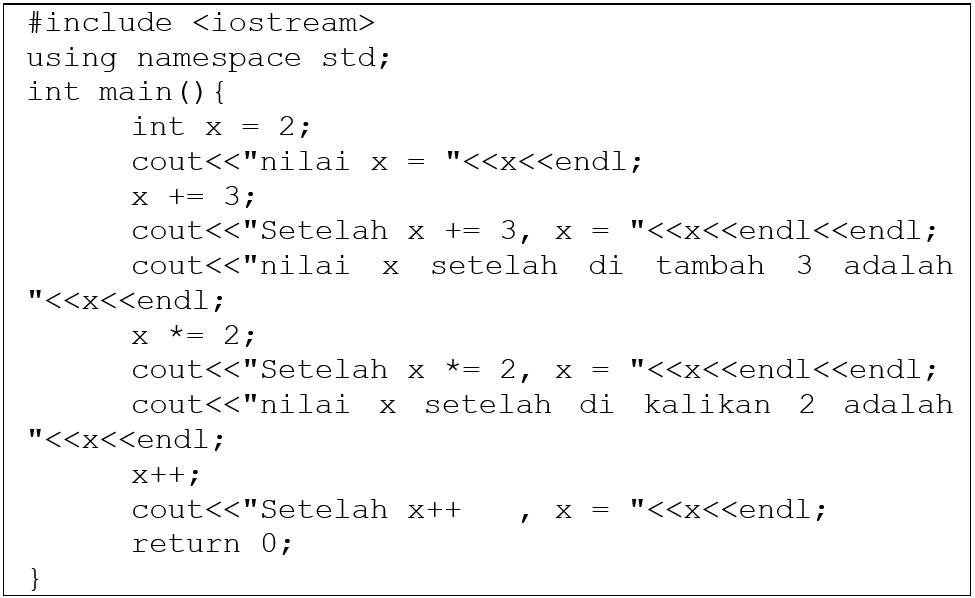
Menjadi :

|  |
| --- |
| x += 2;  y \*= 4; |

|  |  |  |
| --- | --- | --- |
| Operator | Keterangan | Contoh Penggunaanya |
| += | Penjumlahan | z += x sama  dengan, z = z + x |
| -= | Pengurangan | z -= x sama  dengan, z = z - x |
| \*= | Perkalian | z = x sama  dengan, z = z x |
| /= | Pembagian | z /= x sama  dengan, z = z / x |

**Latihan!**

1. Buatlah output dari program di bawah ini beserta analisa program.



### Operator Bitwise (Logika)

Operator logika adalah operasi matematika yang mengoprasikan suatu nilai dalam bilangan biner. Operator bitwise akan bekerja dengan mengubah bilangan desimal menjadi bilangan biner, setelah itu akan melakukan operasi berdasarkan operator bitwise yang digunakan dan pada akhirnya nilai tersebut akan diubah menjadi bilangan desimal kembali. Operator ini hanya bisa digunakan pada operand bertipe data int atau char.

1. **Shift Left (<<)**

Operator Bitwise Shift Left (<<) adalah operator yang akan menggeser nilai dalam bentuk bilangan biner ke kiri.

Bentuk Penulisan :

|  |
| --- |
| Nilai << jumlah; |

Contoh : 12 << 1 ?

|  |
| --- |
| 0000 0000 0000 1100 = 12  //menggeser 1 bit ke kiri, Dibagian kanan disisipkan 0, sebanyak bit yang digeser  0000 0000 0001 1000 = 24 |

1. **Shift Right (>>)**

Operator Bitwise Shift Right (>>) adalah operator yang akan menggeser nilai dalam bentuk bilangan biner ke kanan.

Bentuk Penulisan :

|  |
| --- |
| Nilai >> jumlah; |

Contoh : 12 >> 1 ?

|  |
| --- |
| 0000 0000 0000 1100 = 12  //digeser 1 bit ke kanan, Dibagian kiri disisipkan 0, sebanyak bit yang digeser dan pada bagian paling kanan akan hilang sebanyak bit yang di geser  0000 0000 0000 0110 = 6 |

1. **AND (&)**

Membandingkan berdasarkan sifat dari AND yaitu untuk menghasilkan nilai 1 (true) kedua operand harus bernilai 1(true) jika tidak akan menghasilkan nilai 0 (false).

Pertama, ubah bilangan desimal menjadi biner, dan setelah perbandingan selesai dilakukan, hasil dari perbandingan akan diubah ke bentuk bilangan desimal lagi.

contoh : 12 & 10 ?

|  |  |  |
| --- | --- | --- |
| Operand 1 (12) | Operand 2 (10) | Hasil |
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Hasil : 1000 = 8

1. **OR ( | )**

Membandingkan berdasarkan sifat dari OR yaitu untuk mendapatkan nilai 1 (true) maka salah satu atau semua operand harus bernilai 1 (true), jika semua operand bernilai 0 (false) maka akan mendapatkan nilai 0 (false).

Pertama, ubah bilangan desimal menjadi biner, dan setelah perbandingan selesai dilakukan, hasil dari perbandingan akan diubah ke bentuk bilangan desimal lagi.

contoh : 12 **|** 10 ?

|  |  |  |
| --- | --- | --- |
| Operand 1 (12) | Operand 2 (10) | Hasil |
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Hasil : 1110 = 14

1. **XOR ( ^ )**

Membandingkan berdasarkan sifat dari XOR yaitu untuk mendapatkan nilai 1 (true) maka kedua operand harus memiliki nilai yang berbeda, jika kedua operand memiliki nilai yang sama maka akan mendapatkan nilai 0 ( false ).

Pertama, ubah bilangan desimal menjadi biner, dan setelah perbandingan selesai dilakukan, hasil dari perbandingan akan diubah ke bentuk bilangan desimal lagi.

contoh :12 ^ 10 ?

|  |  |  |
| --- | --- | --- |
| Operand 1 (12) | Operand 2 (10) | Hasil |
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Hasil : 0110 = 6

1. **NOT ( ~ )**

Operator Bitwise NOT dilambangkan dengan menggunakan simbol "~", Operator ini digunakan untuk membalik nilai bit dari suatu operand. Berikut ilustrasi penggunaan Operator Bitwise NOT.

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  int x = 15;  cout<<"~x = " << ~x << endl;  return 0;  } |

Contoh : ~12?

|  |  |
| --- | --- |
| Operand | Hasil |
| 1 | 0 |
| 0 | 1 |

Hasil : (12 + 1) \* -1 = -13*(second complimentary)*

1. **Logika AND ( && )**

Operator AND akan menghasilkan nilai 1 (true) jika semua operand-nya bernilai true. Namun, jika salah satu operand-nya bernilai 0 (false) maka akan menghasilkan nilai 0 (false).

1. **Logika OR ( || )**

Operator OR akan menghasilkan nilai 1 (true), jika salah satu operand-nya bernilai true, dan hanya akan menghasilkan nilai 0 (false) jika seluruh operand-nya bernilai 0 (false).

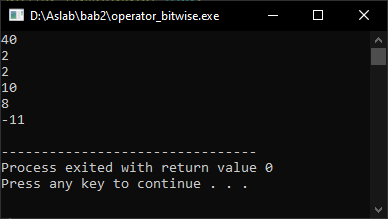
1. **Logika NOT ( ! )**

Operator NOT akan menghasilkan nilai kebalikan dari operand-nya, misal, jika operand-nya bernilai 1 (true) maka akan menghasilkan nilai 0 (false).

Contoh program operator bitwise :

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  int x = 10, y = 2;  cout<<(x<<y)<<endl;  cout<<(x>>y)<<endl;  cout<<(x&y)<<endl;  cout<<(x|y)<<endl;  cout<<(x^y)<<endl;  cout<<(~x)<<endl;  return 0;  } |

Tampilan :



Analisa :

Untuk menghitung hasil dari (and, or, xor, not) output di atas harus di pahami bilangan biner dan gerbang logika.

**Latihan!**

* + 1. Hitunglah :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a. | 8 >> 3 = | c. | 14 ^ 3 = | e. | ~ 567 |
| b. | 6 << 4 = | d. | 7 & 3 = |

### Operator Ternary

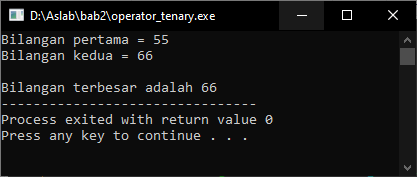
Operator ternary adalah operator yang digunakan dalam operasi yang melibatkan tiga buah operand. Adapun operator yang digunakan untuk menyatakannya adalah operator ?. berikut bentuk umum penggunaan operator ternary.

Kondisi ? Ekspresi2 : Ekspresi3;

Contoh :

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  int x = 55, y = 66;  int max = x > y ? x : y;  cout << "Bilangan pertama = " << x << endl;  cout << "Bilangan kedua = " << y << endl << endl;  cout << "Bilangan terbesar adalah " << max;  return 0;  } |

Tampilan :



Analisa :

Cara menggunakan operator ternary adalah menentukan kondisi dengan bantuan operator hubungan. Pada contoh diatas adalah pada “x>y”. Kemudian tambahkan simbol “?”. Kemudian aksi yang akan muncul pada x:y. Sehingga membentuk :

|  |
| --- |
| max=operand1<operand2?jika\_benar:jika\_salah; |

## Operator I/O

Pada C++ terdapat 2 jenis I/O dasar, yaitu :

1. Statemen Input adalah Statemen / fungsi yang digunakan untuk menerima data dari *keyboard.* Operator yang digunakan “>>”, contoh: cin>> (character in).
2. Statemen Output adalah Statemen yang digunakan untuk menampilkan data ke layar monitor. Operator yang digunakan “<<”, contoh: cout<< (character out).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#taugaksih**  **Identifier**  Identifier atau dalam bahasa Indonesia merupakan Identitas. Identifier adalah identitas atau nama yang telah diberikan kepada function, variabel, obyek, class, namespace dan lain-lain.  Identifier merupakan suatu identitas untuk sebuah deklarasi yang kita dirikan agar CPU, programmer maupun manusia dapat mudah mengenali deklarasi dalam kode program. Di bawah ini adalah macam-macam identifier yang tidak bisa dipakai karena sudah menjadi keyword dalam standar pustaka bahasa pemrograman C++.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | auto | break | case | char | continue | | default | do | double | else | enum | | extern | float | for | goto | If | | int | long | register | short | Signed | | sizeof | static | struct | switch | typedef | | union | unsigned | void | volatile | while | |