



**PROGRAM STUDI  
TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS DIAN NUSWANTORO**

Mata Kuliah  
**Dasar Pemrograman**



# Input/Output, Operasi, Ekspresi

TIM DASAR PEMROGRAMAN  
TEKNIK INFORMATIKA S1  
UNIVERSITAS DIAN NUSWANTORO

## Capaian Pembelajaran

- Setelah mengikuti kuliah ini mahasiswa dapat menjelaskan dan mempraktekkan input/output, operasi, dan ekspresi pada pemrograman procedural dengan diterapkannya pada aksi sekuensial pada permasalahan komputasional sederhana



**FAKULTAS ILMU KOMPUTER  
UNIVERSITAS DIAN NUSWANTORO**

# Input/Output

# Input

- Dengan pustakan iostream

```
cin >> a_int;
```

```
cin >> a_float;
```

```
cin >> a_char;
```

```
cin >> a_int >> a_float;
```

- Dengan pustaka stdio.h

```
scanf("%d", &a_int);
```

```
scanf("%f", &a_float);
```

```
scanf("%c",&a_char);
```

```
scanf("%d%f", &a_int,&a_float);
```

# Output

- Dengan pustakan iostream

```
cout << a_int;
```

```
cout << "Hasilnya adalah:" << a_int;
```

```
cout << a_int << a_float;
```

- Dengan pustaka stdio.h

```
printf("%d", a_int);
```

```
printf("%f", a_float);
```

```
printf("Hasil lebih presisi: %2f",  
a_float); printf("%c",a_char);
```

```
printf("%d%f", a_int,a_float);
```

# Format input/output pada pustaka stdio.h

| Data Type              | Format Specifier |
|------------------------|------------------|
| int                    | %d               |
| char                   | %c               |
| float                  | %f               |
| double                 | %lf              |
| short int              | %hd              |
| unsigned int           | %u               |
| long int               | %li              |
| long long int          | %lli             |
| unsigned long int      | %lu              |
| unsigned long long int | %llu             |
| signed char            | %c               |
| unsigned char          | %c               |
| long double            | %Lf              |

# Operator Aritmatika

- Digunakan untuk melakukan operasi matematis terhadap suatu nilai. Operator ini paling sering digunakan dalam implementasi hampir di seluruh program yang ada.

| No | Simbol | Nama            | Contoh         |
|----|--------|-----------------|----------------|
| 1  | +      | Penjumlahan     | Hasil = 3 + 10 |
| 2  | -      | Pengurangan     | Hasil = 10 – 7 |
| 3  | *      | Perkalian       | Hasil = 9 * 3  |
| 4  | /      | Pembagian       | Hasil = 10 / 2 |
| 5  | %      | Sisa Hasil bagi | Hasil = 10 % 3 |

**NB = Urutan prioritas operatornya adalah ( \* -> / -> % lalu + -> - )**

# Contoh Operator Aritmatika - 1

- ❑ Contoh dibawah merupakan ekspresi sederhana dari operator aritmatika.
- ❑ Dalam 1 ekspresi dapat mengandung **lebih dari** 1 operator
- ❑ Urutan eksekusi perhitungan operator dapat dilihat pada slide sebelumnya.

```
int hasil;  
  
hasil = 10 + 3;
```

```
int operand_1 = 30;  
int operand_2 = 50;  
int hasil;  
  
hasil = operand_1 - operand_2 + 100;
```



## Contoh Operator Aritmatika - 2

```
int hasil;  
hasil = (88 + 12) / 5;
```

```
int hasil;  
hasil = 60 + 30 * 7;
```

- ❑ Contoh disamping merupakan penggunaan tanda ***kurung*** sebagai perintah untuk memprioritaskan secara paksa tanpa terikat oleh aturan urutan eksekusi operator.
- ❑ Misalkan operator + dapat dihitung dahulu daripada bagi dengan memberikan tanda kurung.
- ❑ Apabila tidak ada tanda kurung , maka urutan prioritas operator akan disesuaikan secara default

```
float hasil;  
float angka_1 = 10;  
float angka_2 = 4;  
  
hasil = (88 + 12) / 5 * (10 - angka_1) * angka_2 + 11 / 3 + 100;
```

# Operator Perbandingan

- ❑ Operator perbandingan atau relasional digunakan untuk melakukan perbandingan terhadap 2 nilai operand, umumnya hasil yang diberikan dari operator relasional berupa nilai Boolean (**TRUE** atau **FALSE**).

| No | Simbol | Nama                            | Contoh              |
|----|--------|---------------------------------|---------------------|
| 1  | >      | Apakah lebih besar ?            | Hasil = ( 3 > 10 )  |
| 2  | <      | Apakah lebih kecil ?            | Hasil = ( 10 < 7 )  |
| 3  | >=     | Apakah lebih besar sama dengan? | Hasil = ( 9 >= 3 )  |
| 4  | <=     | Apakah lebih kecil sama dengan? | Hasil = ( 10 <= 3 ) |
| 5  | ==     | Apakah sama ?                   | Hasil = ( 10 == 4 ) |
| 6  | !=     | Apakah tidak sama ?             | Hasil = ( 10 != 3 ) |

**NB = Urutan prioritas operatornya adalah dari KIRI ke KANAN**

# Contoh Operator Relasional - 1

- Contoh dibawah merupakan ekspresi sederhana dari operator relasional

```
int angka_1 = 20;  
int angka_2 = 5;  
bool hasil_relasi;  
  
hasil_relasi = (angka_1 == angka_2);
```

- Pada contoh di kiri operator “==” berguna untuk mengecek apakah kedua operand memiliki nilai yang sama ? , jika sama maka hasil **TRUE** selain itu **FALSE**

- Pada contoh di kanan menggunakan 3 operator . Urutan prioritas akan dieksekusi dari kiri ke kanan , sehingga akan menghasilkan nilai 1 (**TRUE**)

```
bool hasil_relasi;  
  
hasil_relasi = -100 > 3 < 4;
```

- - 100 > 3 < 4
- FALSE < 4
- 0 < 4
- True
- 1

# Operator Logika

- ❑ Operator Logika berfungsi untuk melakukan operasi yang berkaitan dengan tipe data Boolean . Baik operatornya maupun operandnya. Hasil operasinya berupa nilai Boolean.

| No | Simbol | Nama                  | Contoh                       |
|----|--------|-----------------------|------------------------------|
| 1  | &&     | AND                   | Hasil = TRUE && FALSE        |
| 2  |        | OR                    | Hasil = (3 > 10)    (10 < 4) |
| 3  | !      | NOT (Negasi/Ingkaran) | Hasil = ! (10 == 3)          |
| 4  | ^      | XOR (Exlusive OR)     | Hasil = 1 ^ 0                |

**NB = Urutan prioritas operatornya adalah ( XOR -> && -> || )**

# Contoh Operator Logika - 1

```
bool operand_1 = true;  
bool operand_2 = false;  
  
bool hasil_logika;  
  
hasil_logika = operand_1 && operand_2;
```

- ❑ Contoh sederhana dari operator logika, dengan operator **AND**.
- ❑ Dapat diisi lebih dari 1 operator juga

```
bool hasil_logika;  
  
hasil_logika = (false && true) ^ true ;
```

- ❑ Penggunaan juga berlaku untuk operator logika untuk mendahulukan operasi perhitungan.

## Gabungan dari penggunaan operator logika dan relasional

```
bool hasil_logika;  
  
hasil_logika = (false && true) ^ true || (100 < 90) && (!false) ^ (90==90) ;
```

## Penyederhanaan Ekspresi Aritmatika

| No | Simbol | Nama                             | Contoh      |
|----|--------|----------------------------------|-------------|
| 1  | =      | Mengisi nilai dari kanan ke kiri | Nilai = 4   |
| 2  | +=     | Setara dengan $C = C + A$        | Nilai += 10 |
| 3  | -=     | Setara dengan $C = C - A$        | Nilai -= 3  |
| 4  | *=     | Setara dengan $C = C * A$        | Nilai *= 6  |
| 5  | /=     | Setara dengan $C = C / A$        | Nilai /= 4  |
| 6  | %=     | Setara dengan $C = C \% A$       | Nilai %= 7  |

# Increment vs Decrement

- Misal memiliki variabel `i` dengan nilai awal 0
  - `i = i + 1;` {artinya `i` akan naik nilainya dari 0 menjadi 1, sehingga di sebut dengan isitilah increment}
  - Bisa ditulis dengan `i++;`
- Misal memiliki variabel `i` dengan nilai awal 10
  - `i = i - 1;` {artinya `i` akan turun nilainya dari 10 menjadi 9, sehingga di sebut dengan isitilah decrement}
  - Bisa ditulis dengan `i--;`
- Dibahasa C/C++ increment atau decrement bisa juga di tulis dengan `++i` atau `--i`, tetapi ada perbedaan.

## i++ VS ++i

- Contoh ++i
  - Misal kita memiliki variabel  $i = 1$
  - Kita ingin men assign ++i kedalam variabel j dengan cara  $j = ++i$ ;
  - Berapa nilai i dan berapa nilai j setelah proses assignment tersebut terjadi?
  - Nilai i akan menjadi 2 dan nilai j akan menjadi 2. Penjabaran:
    - Proses pertama nilai i akan mengalami increment sehingga dia akan menghasilkan nilai 2, proses kedua hasil 2 tadi diassign ke j
- Contoh i++
  - Misal kita memiliki variabel  $i = 1$
  - Kita ingin men assign i++ kedalam variabel j dengan cara  $j = i++$ ;
  - Berapa nilai i dan berapa nilai j setelah proses assignment tersebut terjadi?
  - Nilai i akan menjadi 2 dan nilai j akan menjadi 1. Penjabaran:
    - Proses pertama nilai i akan diassign ke j terlebih dahulu sehingga nilai j adalah 1, proses kedua increment pada variabel i baru dilakukan setelahnya sehingga hasilnya adalah nilai i menjadi 2.



## Bentuk operator penyederhanaan

- Contoh:  $i = i * 2$  bisa ditulis  $i *= 2$ ;
- Contoh:  $i = i + 2$  bisa ditulis  $i += 2$ ;
- Contoh:  $i = i - 10$  bisa ditulis  $i -= 10$ ;
- Contoh:  $i = i / 5$  bisa ditulis  $i /= 5$ ;
- dll

# Program Sequential1

## Program Sequential1

{Contoh penulisan aksi sekuensial dengan memanfaatkan apa yang sudah dipelajari sebelumnya}

### KAMUS

i : integer  
x : real  
hasil : integer

### ALGORITMA

input(i)  
x  $\leftarrow$  12.5  
hasil  $\leftarrow$  i \* 10  
output(i)  
output(x+5.4)  
output(i,x,hasil)

# Program Sequential1 dengan C++

```
main.cpp x
1  /*
2  JUDUL: Program Sequential1
3  Contoh penulisan aksi sekuensial dengan memanfaatkan apa yang sudah dipelajari sebelumnya
4  */
5  #include <iostream> //bisa diganti #include <stdio.h>
6
7  using namespace std; //hapus jika menggunakan stdio.h
8
9  int main()
10 {
11     //Kamus
12     int i;
13     float x;
14     int hasil;
15
16     //Algoritma
17     cin >> i; // jika menggunakan stdio.h diganti scanf("%d",&i);
18     x = 12.5;
19     hasil = i * 10;
20     cout << i << endl; // jika menggunakan stdio.h diganti printf("%d \n",i)
21     cout << (x+5.4) << endl;
22     cout << i << x << hasil << endl;
23
24
25     return 0;
26 }
```

# Program PersegiPanjang

**Program PersegiPanjang**  
{program untuk menghitung keliling dan luas persegi panjang}

Coba buat dengan bahasa c++?

## KAMUS

p : real  
l : real  
hasilKeliling : real  
hasilLuas : real

## ALGORITMA

input(p)  
input(l)  
hasilKeliling  $\leftarrow 2 * (p+l)$   
hasilLuas  $\leftarrow p*l$   
output(hasilKeliling)  
output(hasilLuas)

# Tugas Tambahan untuk di Lab

## Membuat Tabel Operator Logika

| a | b | a <u>and</u> b | a <u>or</u> b | <u>not</u> a | <u>not</u> b | a <u>eq</u> b | a <u>neq</u> b |
|---|---|----------------|---------------|--------------|--------------|---------------|----------------|
| T | T | T              | T             | F            | F            | T             | F              |
| T | F | F              | T             | F            | T            | F             | T              |
| F | T | F              | T             | T            | F            | F             | T              |
| F | F | F              | F             | T            | T            | T             | F              |

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      bool a1=1, a2=1, a3=0, a4=0;
8      bool b1=1, b2=0, b3=1, b4=0;
9
10     cout << "-----" << endl;
11     cout << "| a | b | a and b | a or b | not a | not b | a eq b | a neq b |\n";
12     cout << "-----" << endl;
13     cout << "| " << a1 << " | " << b1 << " | " << (a1&b1) << " | " << (a1||b1) << " | " <<
14     (!a1) << " | " << (!b1) << " | " << (a1==b1) << " | " << !(a1==b1) << " |\n";
15     cout << "| " << a2 << " | " << b2 << " | " << (a2&b2) << " | " << (a2||b2) << " | " <<
16     (!a2) << " | " << (!b2) << " | " << (a2==b2) << " | " << !(a2==b2) << " |\n";
17
18     //operator ==, !=, >, >=, <, <=
19     return 0;
20 }

```

# Referensi

## Utama:

1. Bjarne Stroustrup, 2014, Programming: Principles and Practice Using C++ (Second Edition), Addison-Wesley Professional

## Pendukung:

1. Introduction to Computer Science and Programming in Python, MIT  
<https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016>
2. Introduction to Computer Science and Programming, MIT  
<https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00sc-introduction-to-computer-science-and-programming-spring-2011/index.htm>



# TERIMA KASIH

ANY QUESTIONS?