STRUKTUR DATA

#04 – User Input dan Operator

Oleh: Fahmi Ruziq, S.T., M.Kom.

User Input

User Input

Pada dasarnya, **input pengguna** di C++ adalah cara sebuah program menerima data yang dimasukkan oleh pengguna saat program berjalan. Ini memungkinkan program menjadi interaktif dan dinamis, tidak hanya menampilkan output yang telah ditentukan sebelumnya. Input ini bisa berupa angka, teks, atau karakter, dan program akan memprosesnya untuk melakukan tugas-tugas tertentu.

Menggunakan `cin` untuk Input

Di C++, kita menggunakan objek `cin` **(Console Input)** yang merupakan bagian dari `iostream` untuk membaca data dari **keyboard**. `cin` bekerja dengan operator ekstraksi `>>`. Objek `cin` ini akan membaca data dari *stream* input standar (biasanya keyboard) dan menyimpannya ke dalam sebuah variabel.

Sintaks Dasar

Sintaksnya sangat sederhana:

`cin >> umur;` akan menunggu pengguna memasukkan angka dan menyimpannya ke variabel umur.

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

int main() {
    int umur;
    cout << "Masukkan umur Anda: ";
    cin >> umur;
    cout << "Umur Anda adalah: " << umur << " tahun." << endl;

    string nama;
    cout << "Masukkan nama Anda: ";
    cin >> nama;
    cout << "Halo, " << nama << "!" << endl;

    return 0;
}
```

Sintaks Dasar

Sintaksnya sangat sederhana:

`cin >> nama;` akan membaca kata pertama yang dimasukkan pengguna (hingga spasi atau baris baru) dan menyimpannya ke variable nama.

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

int main() {
    int umur;
    cout << "Masukkan umur Anda: ";
    cin >> umur;
    cout << "Umur Anda adalah: " << umur << " tahun." << endl;

    string nama;
    cout << "Masukkan nama Anda: ";
    cin >> nama;
    cout << "Halo, " << nama << "!" << endl;

    return 0;
}
```

Masalah dengan Spasi pada Input Teks

Salah satu tantangan umum saat menggunakan `cin` untuk string adalah cara kerjanya. Operator `>>` akan berhenti membaca saat menemui **karakter spasi (whitespace)**. Ini berarti jika pengguna memasukkan “Budi Santoso", `cin >> nama;` hanya akan mengambil “Budi", dan “Santoso" akan diabaikan.

Mengatasi Masalah Spasi dengan `getline()`

Untuk membaca seluruh baris, termasuk spasi, kita bisa menggunakan fungsi `getline()`. Berikut adalah cara penggunaannya:

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

int main() {
    string namaLengkap;
    cout << "Masukkan nama lengkap Anda: ";
    getline(cin, namaLengkap);
    cout << "Nama lengkap Anda adalah: " << namaLengkap << endl;
    return 0;
}
```

Mengatasi Masalah Spasi dengan `getline()`

Dalam contoh ini, `getline(cin, namaLengkap);` akan membaca seluruh baris yang diketik pengguna, termasuk spasi, hingga pengguna menekan tombol Enter.

Mengatasi Newline Character (\n)

Ketika kita menggabungkan `cin` untuk angka (atau input lain yang bukan string) dengan `getline()`, kita bisa menghadapi masalah. Setelah `cin` membaca angka, karakter *newline* (`\n`) yang dihasilkan saat pengguna menekan Enter **masih tersisa** di *input buffer*.

Mengatasi Newline Character (\n)

Contoh masalah:

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

int main() {
    int usia;
    string hobi;

    cout << "Masukkan usia Anda: ";
    cin >> usia;

    cout << "Masukkan hobi Anda: ";
    getline(cin, hobi); // getline akan membaca \n yang tersisa

    cout << "Usia: " << usia << ", Hobi: " << hobi << endl;
    return 0;
}
```

Mengatasi Newline Character (`\n`)

Dalam kode di atas, `getline()` akan segera membaca *newline character* yang tersisa dari input usia, sehingga pengguna tidak akan sempat memasukkan hobi.

Solusi: Menggunakan `cin.ignore()`

Untuk mengatasi masalah ini, kita perlu membersihkan *input buffer* sebelum memanggil `getline()`. Kita bisa menggunakan `cin.ignore()` untuk mengabaikan karakter yang tersisa.

Mengatasi Newline Character (\n)

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

int main() {
    int usia;
    string hobi;

    cout << "Masukkan usia Anda: ";
    cin >> usia;

    // Membersihkan buffer hingga newline
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Masukkan hobi Anda: ";
    getline(cin, hobi); // getline akan membaca \n yang tersisa

    cout << "Usia: " << usia << ", Hobi: " << hobi << endl;
    return 0;
}
```

Mengatasi Newline Character (\n)

Penjelasan `cin.ignore()`:

`numeric_limits<streamsize>::max()`: Mengabaikan jumlah karakter maksimum yang mungkin ada di buffer.

`'\n'`: Berhenti mengabaikan saat menemukan karakter newline.

Dengan cara ini, `getline()` akan mulai membaca dari *buffer* yang sudah bersih, memungkinkan pengguna untuk memasukkan teks dengan benar.

Operator

Operator

Dalam bahasa pemrograman C++, **operator** adalah simbol yang digunakan untuk melakukan operasi pada satu atau lebih nilai atau variabel. Nilai atau variabel ini disebut **operand**. Memahami operator sangat penting karena operator memungkinkan kita untuk memanipulasi data dan melakukan perhitungan.

Jenis-Jenis Operator

Operator ini digunakan untuk melakukan operasi matematika dasar.

1. Operator Aritmatika

Operator ini digunakan untuk melakukan operasi matematika dasar.

Operator	Deskripsi	Contoh	Hasil
+	Penjumlahan	5 + 2	7
-	Pengurangan	5 - 2	3
*	Perkalian	5 * 2	10
/	Pembagian	10 / 5	2
%	Sisa Pembagian	5 % 2	1

Jenis-Jenis Operator

Catatan: Operator `/` akan menghasilkan nilai *integer* (bilangan bulat) jika kedua operand-nya adalah *integer*. Untuk mendapatkan hasil desimal, setidaknya salah satu operand harus berupa *floating-point* (misalnya, `5.0 / 2`).

Jenis-Jenis Operator

2. Operator Penugasan (Assignment)

Operator ini digunakan untuk memberikan nilai kepada variabel.

Operator	Contoh	Sama dengan
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5

Jenis-Jenis Operator

3. Operator Perbandingan (Relational)

Operator ini digunakan untuk membandingkan dua nilai dan akan menghasilkan nilai boolean (`true` atau `false`).

Operator	Deskripsi	Contoh	Sama dengan
==	Sama dengan	5 == 2	FALSE
!=	Tidak sama dengan	5 != 2	TRUE
>	Lebih besar dari	5 > 2	TRUE
<	Lebih kecil dari	5 < 2	FALSE
>=	Lebih besar dari atau sama dengan	5 >= 2	TRUE
<=	Lebih kecil dari atau sama dengan	5 <= 2	FALSE

Jenis-Jenis Operator

4. Operator Logika (Logical)

Operator ini digunakan untuk menggabungkan atau memanipulasi ekspresi boolean.

Operator	Deskripsi	Contoh	Sama dengan
&&	Logika AND	(5 > 2) && (5 < 10)	TRUE
||	Logika OR	(5 < 2) || (5 < 10)	TRUE
!	Logika NOT	!(5 == 2)	TRUE

Jenis-Jenis Operator

5. **Operator Aritmatika Unary (Increment dan Decrement)**

   Operator ++ dan −− digunakan untuk menaikkan atau menurunkan nilai variabel sebesar satu.

   ++ (**Increment**): Menambah nilai variabel sebesar 1.

   −− (**Decrement**): Mengurangi nilai variabel sebesar 1.

Jenis-Jenis Operator

Ada dua versi dari operator ini:

Prefix: Operator diletakkan sebelum variabel (++x). Nilai variabel diubah sebelum digunakan dalam ekspresi.

Postfix: Operator diletakkan setelah variabel (x++). Nilai variabel diubah setelah digunakan dalam ekspresi.

Contoh:

```
int a = 10;
int b = 10;

int c = ++a; // c = 11, a = 11
int d = b++; // d = 10, b = 11
```

Contoh Program Lengkap

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

int main() {
    // 1. Operator Aritmatika
    int a = 10;
    int b = 3;

    cout << "--- Operator Aritmatika ---" << endl;
    cout << "a + b = " << a + b << endl;
    cout << "a - b = " << a - b << endl;
    cout << "a * b = " << a * b << endl;
    cout << "a / b = " << a / b << endl; // Hasil integer
    cout << "a % b = " << a % b << endl;
    cout << endl;
```

Contoh Program Lengkap

```
// 2. Operator Penugasan
int x = 5;
cout << "--- Operator Penugasan ---" << endl;
cout << "Nilai awal x: " << x << endl;
x += 3; // x = x + 3
cout << "x setelah += 3: " << x << endl;
cout << endl;

// 3. Operator Perbandingan
int p = 8;
int q = 5;
cout << "--- Operator Perbandingan ---" << endl;
cout << "p == q: " << (p == q) << endl;
cout << "p != q: " << (p != q) << endl;
cout << "p > q: " << (p > q) << endl;
cout << "p < q: " << (p < q) << endl;
cout << endl;
```

Contoh Program Lengkap

```
    // 4. Operator Logika
    bool kondisi1 = (p > 5); // true
    bool kondisi2 = (q < 5); // false
    cout << "--- Operator Logika ---" << endl;
    cout << "kondisi1 && kondisi2: " << (kondisi1 && kondisi2) << endl;
    cout << "kondisi1 || kondisi2: " << (kondisi1 || kondisi2) << endl;
    cout << "!kondisi1: " << (!kondisi1) << endl;
    cout << endl;

    // 5. Operator Increment/Decrement
    int counter = 5;
    cout << "--- Operator Increment/Decrement ---" << endl;
    cout << "Nilai awal counter: " << counter << endl;
    cout << "Prefix Increment (++counter): " << ++counter << endl; // counter menjadi 6
    cout << "Nilai counter sekarang: " << counter << endl;
    cout << "Postfix Increment (counter++): " << counter++ << endl; // cetak 6, lalu counter menjadi 7
    cout << "Nilai counter setelahnya: " << counter << endl;

    return 0;
}
```

Terima kasih.