



Masukan/Keluaran

Tim Olimpiade Komputer Indonesia

Bagian 1

Masukan



Kilas Balik: kuadrat.cpp

- Sekarang coba lihat kembali program kuadrat.cpp:

```
#include <stdio>
```

```
int a, b, c, x, hasil;
```

```
int main() {
```

```
    a = 1;
```

```
    b = 3;
```

```
    c = -2;
```

```
    x = 2;
```

```
    hasil = a*x*x + b*x + c;
```

```
    printf("ax^2 + bx + c = %d\n", hasil);  
}
```

- Jika kita ingin mengganti nilai x, kode harus diganti, dikompilasi ulang, baru dijalankan kembali.
- Untuk menghasilkan keluaran yang bervariasi, perlu ada masukan dari luar program.



Membaca Masukan

- Diperlukan mekanisme untuk melakukan pembacaan masukan dari luar program.
- Masukan bagi suatu program bisa berasal dari berbagai sumber, misalnya *standard input* atau *file*.
- Kita akan mempelajari fungsi yang umum untuk membaca masukan, yaitu **scanf**.



Membaca Masukan: scanf

- Modifikasi bagian `x = 2` menjadi `scanf("%d", &x):`

```
#include <stdio>
```

```
int a, b, c, x, hasil;
```

```
int main() {
```

```
    a = 1;
```

```
    b = 3;
```

```
    c = -2;
```

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

```
    hasil = a*x*x + b*x + c;
```

```
    printf("ax^2 + bx + c = %d\n", hasil);
```

```
}
```

- Kompilasi, dan jalankan program. Kemudian ketikkan angka 2, dan tekan enter.
- Selamat! Kalian berhasil membaca masukan!



Fungsi scanf

- Fungsi **scanf** berguna untuk membaca masukan, dan nilainya dapat di-*assign* ke dalam variabel.
- Fungsi ini disediakan oleh STL cstdio.
- Cara kerja scanf: pada berkas masukan, cari *token* yang dapat dibaca berikutnya, lalu baca ambil nilainya.
- Yang dimaksud *token* adalah serangkaian karakter non-spasi, misalnya huruf atau angka.
- Pada contoh sebelumnya, *token* yang dimaksud adalah bilangan yang akan menjadi nilai variabel *x*.



Fungsi scanf (lanj.)

- Sama dengan `printf`, diperlukan simbol sesuai tipe data yang bersangkutan.
- Perbedaan paling mendasar adalah diperlukannya karakter `'&'` pada variabel yang hendak diisi.



Membaca Beberapa Variabel

- Hal ini juga berlaku apabila Anda hendak membaca beberapa variabel pada satu baris masukan.

```
#include <stdio>
```

```
int a, b, c, x, hasil;
```

```
int main() {  
    scanf("%d %d %d %d", &a, &b, &c, &x);  
  
    hasil = a*x*x + b*x + c;  
    printf("ax^2 + bx + c = %d\n", hasil);  
}
```

- Jalankan program lalu masukkan "1 3 -2 2", lalu tekan enter.



Membaca Beberapa Variabel (lanj.)

- Meskipun kita memberikan pola "%d %d %d %d", tidak masalah apabila masukan yang hendak Anda baca ada di baris yang berbeda.
- Misalnya Anda dapat ketikkan "1 3", enter, "-2 2", lalu enter.
- Masukan tetap akan dibaca sesuai urutan yang diberikan.
- Alasannya adalah scanf membaca bilangan dengan cara mencari *token* yang ada selanjutnya, tanpa peduli baris baru atau spasi.



Membaca Karakter

- Terkecuali pada tipe data karakter, scanf **tidak** membaca *token* selanjutnya.
- Scanf akan membaca 1 karakter selanjutnya, baik itu spasi, angka, ataupun baris baru.



Membaca Karakter (lanj.)

- Perhatikan contoh program berikut.

```
#include <stdio>
```

```
char c1, c2, c3;  
int bil;
```

```
int main() {  
    scanf("%c %c", &c1, &c2);  
    scanf("%d", &bil);  
    scanf("%c", &c3);  
  
    printf("c1='%c' c2='%c' bil=%d c3='%c'\n", c1, c2,  
        bil, c3);  
}
```

- Berikan masukan berupa "p q", enter, 5, enter, lalu "r".



Membaca Karakter (lanj.)

- Berikut adalah keluaran yang dihasilkan:

c1='p' c2='q' bil=5 c3='
,

- Perhatikan bahwa c3 memiliki nilai berupa karakter enter, padahal yang kita harapkan adalah karakter 'r'.
- Hal ini disebabkan karena karakter yang selanjutnya dimasukkan sesudah membaca bil adalah enter, yang kemudian dibaca untuk c3.



Membaca Karakter (lanj.)

- Cara yang tepat adalah dengan menambahkan "\n" secara tertib di akhir pembacaan baris:

```
#include <stdio>
```

```
char c1, c2, c3;  
int bil;
```

```
int main() {  
    scanf("%c %c\n", &c1, &c2);  
    scanf("%d\n", &bil);  
    scanf("%c", &c3);  
  
    printf("c1='%c' c2='%c' bil=%d c3='%c'\n", c1, c2,  
        bil, c3);  
}
```

- Karena berpotensi membingungkan dan memperumit penulisan kode, pembacaan tipe data karakter kurang disarankan.



Membaca String

- Cara yang disarankan adalah membacanya dalam bentuk string, sekalipun yang akan dibaca dipastikan hanya memiliki 1 karakter.
- Seperti printf, scanf tidak dapat berinteraksi secara langsung dengan STL string.
- Scanf perlu membaca string dalam bentuk cstring, kemudian mengubahnya menjadi string.



Membaca String (lanj.)

- Perhatikan program berikut:

```
#include <stdio>
#include <string>

using namespace std;

char buff[1001];

int main() {
    scanf("%s", buff);

    string s = buff;
    printf("s='%s'\n", s.c_str());
}
```

- Variabel buff merupakan *array of char* dengan maksimal 1001 karakter (angka ini dapat Anda ubah sesuai kebutuhan).
- Array of char* inilah yang merupakan cstring.



Membaca String (lanj.)

- Kita dapat membaca cstring dengan scanf, lalu mengubahnya ke bentuk string dengan melakukan *assignment* ke variabel string (`string s = buff`).
- Khusus untuk pembacaan cstring, tanda '&' tidak digunakan.
- Program tersebut akan membaca 1 *token* yang diberikan.
- Coba jalankan program tersebut, lalu masukkan "abcd", lalu enter.



Membaca Sebaris String

- Bagaimana jika kita hendak membaca sebuah baris string, yang mungkin mengandung spasi?
- Caranya adalah menggunakan simbol khusus "%[^\\n]\\n".

```
#include <stdio>
```

```
#include <string>
```

```
using namespace std;
```

```
char buff[1001];
```

```
int main() {  
    scanf("%[^\\n]\\n", buff);  
  
    string s = buff;  
    printf("s='%s'\\n", s.c_str());  
}
```



Kesimpulan dalam Membaca Masukan

- Membaca masukan pada C++ mungkin tidak semudah yang diharapkan.
- Anda perlu menghafal sintaks dan cara pembacaan bilangan, karakter, dan string.
- Untungnya, hal-hal yang telah diajarkan tersebut sudah cukup untuk membuat program yang kompleks.
- Cobalah untuk mencetak kembali nilai variabel yang telah dibaca, untuk memastikan program Anda membaca masukan dengan tepat!



Bagian 2

Keluaran



Mencetak Keluaran

- Seperti masukan, keluaran juga bisa disajikan dalam bentuk langsung ke *standard output* atau ke *file*.
- Pada C++, fungsi untuk mencetak keluaran yang umum adalah **printf**.
- Sejauh ini, kita sudah menggunakan printf untuk berbagai keperluan dan Anda seharusnya telah menguasainya.



Contoh Program: jumlah.cpp

- Coba ketikkan dan jalankan program berikut:

```
#include <stdio>
```

```
int main() {  
    int a, b;  
    printf("masukkan nilai a: ");  
    scanf("%d", &a);  
    printf("masukkan nilai b: ");  
    scanf("%d", &b);  
    printf("hasil dari penjumlahan a dan b: %d\n", a+b);  
}
```

- Pada program tersebut, dicetak terlebih dahulu apa yang perlu dimasukkan. Tentu saja, program seperti ini sangat ramah terhadap pengguna (*user-friendly*).
- Namun dalam kontes pemrograman OSN/IOI, hal seperti ini tidak perlu dilakukan. Bahkan, tidak boleh dilakukan.



Bagian 3

Standard Input Output



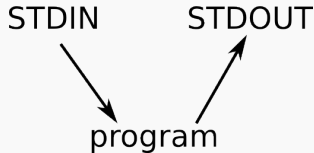
Penjelasan Tentang STDIO

- Tempat kalian selama ini mengisikan masukan dan melihat keluaran biasa disebut sebagai *standard input output*, atau **STDIO**.
- **STDIO** memiliki dua saluran yang berbeda, yaitu *input* (**STDIN**) dan *output* (**STDOUT**).



Penjelasan Tentang STDIO (lanj.)

- Masukan yang kalian masukkan, akan melewati saluran STDIN.
- Keluaran yang kalian lihat, sebenarnya datang lewat saluran STDOUT.
- Namun, pada *command line* keduanya terlihat seperti menyatu, seakan-akan keduanya melewati jalur yang sama.



Penjelasan Tentang STDIO (lanj.)

- Untuk lebih memahami tentang hal ini, coba buat sebuah berkas bernama `input.txt` pada *folder* yang sama dengan program `jumlah.cpp`, dan berisi:

```
1  
2
```

- Kemudian pada *command line*, saat menjalankan program `jumlah.cpp`, ketikkan perintah:

```
jumlah < input.txt > output.txt
```

- Buka `output.txt` dan perhatikan apa yang tercetak!



Penjelasan Tentang STDIO (lanj.)

- Isi dari output.txt adalah:

```
masukkan nilai a:  
masukkan nilai b:  
hasil dari penjumlahan a dan b: 3
```

- Tulisan "masukkan nilai ..." juga ikut tercetak, karena pada kasus ini, **STDOUT** merupakan berkas output.txt. Segala yang dicetak lewat saluran **STDOUT** akan dicetak ke output.txt.
- Dengan pemahaman yang sama, seluruh masukan yang diberikan adalah lewat **STDIN**, yang merupakan input.txt. Sehingga masukannya perlu dimasukkan ke input.txt terlebih dahulu.



Masukan dan Keluaran pada OSN/IOI

- Setelah kalian memahami tentang **STDIN** dan **STDOUT**, mungkin kalian sudah bisa menebak kenapa pada OSN/IOI tidak boleh mencetak informasi masukan seperti "masukkan nilai ...".
- Hal ini dikarenakan tulisan itu akan ikut tercetak sebagai keluaran, yang mana mengakibatkan ada keluaran yang tidak sesuai spesifikasi soal. Hasilnya, program akan dinilai *wrong answer*, alias menghasilkan jawaban yang tidak sesuai.



Selanjutnya...

- Kini kalian sudah mempelajari tentang variabel, ekspresi, dan masukan/keluaran.
- Artinya, sudah waktunya untuk menulis program-program sederhana.

