



Variabel dan Tipe Data

Tim Olimpiade Komputer Indonesia

Pendahuluan

Melalui dokumen ini, kalian akan:

- Mengenal konsep variabel.
- Mempelajari berbagai tipe data.
- Mempelajari cara deklarasi variabel.
- Mengenal operasi assignment.



Kilas Balik

- Mari kita lihat kembali program halo.cpp.

```
#include <stdio>
```

```
int main() {  
    printf("halo dunia\n");  
}
```

- Pada program tersebut, terdapat kata kunci "int main() {" dan "}".
- Kedua kata kunci tersebut blok program utama.
- Ketika halo.cpp dieksekusi, seluruh perintah di blok program utama akan dieksekusi secara berurutan.



Baris Perintah Program

- Pada `halo.cpp`, satu-satunya perintah yang ada adalah `printf("halo dunia\n");`
- Pada C++, `printf(x)` merupakan fungsi untuk mencetak `x` ke layar.
- Dalam program ini, `x = 'halo dunia\n'`.
- `'\n'` merupakan karakter "baris baru" atau "enter".



Bagian 1

Konsep Variabel



Perkenalan Variabel

Variabel

Merupakan istilah yang diadopsi dari dunia matematika, yang memetakan sebuah nama ke suatu nilai.



Perkenalan Variabel (lanj.)

- Setiap kali suatu variabel digunakan dalam ekspresi matematika, yang diacu sebenarnya adalah nilai yang dipetakan oleh nama variabel tersebut.
- Contoh: jika kita menyatakan $x = 5$, maka hasil dari $3x^2 + x$ adalah 80.
- Dalam pemrograman, kita bisa membuat variabel, mengisikan nilai pada variabel, dan mengacu nilai yang dipetakan variabel tersebut.



Aturan Penamaan Variabel

- Variabel bebas diberi nama apapun, tetapi terbatas pada beberapa aturan berikut:
 - Terdiri dari kombinasi karakter huruf, angka, dan underscore (-).
 - Tidak boleh dimulai dengan angka.
 - Huruf kapital dan huruf kecil dianggap berbeda. Artinya "a1" dan "A1" dianggap merupakan dua variabel yang berbeda.
 - Tidak boleh merupakan *reserved word*. Contoh *reserved word* pada C++: **int**, **if**, **while**, **for**, atau **switch**.
- Contoh penulisan variabel yang tepat: nilai, xKecil, y1, tambahan_string.
- Contoh penulisan variabel yang salah: 2kar, wow!?, while.



Aturan Penamaan Variabel (lanj.)

- Lebih jauh lagi, aturan ini berlaku pada seluruh penamaan *identifier*, yaitu nama variabel dan fungsi yang akan dipelajari selanjutnya.



Assignment

Assignment

Pengisian nilai yang diacu oleh variabel dengan suatu nilai disebut *assignment*.

- Operator untuk *assignment* adalah =
- Isikan ruas kiri dengan nama suatu variabel, dan ruas kanan dengan nilai yang ingin diisikan ke variabel tersebut.
- Tipe data dari variabel dan nilai yang diacu **harus sesuai**.



Contoh Program: assign.cpp

- Perhatikan contoh program assign.cpp berikut. Tuliskan, lalu jalankan program ini.

```
#include <stdio>

int x;

int main() {
    x = 12;
    printf("Nilai = %d\n", x);
}
```



Penjelasan Program: assign.cpp

- Keluaran yang dihasilkan dari program itu adalah sebuah baris berisikan:

```
Nilai = 12
```

- Pada program tersebut, `x` merupakan suatu variabel.
- Variabel `x` didaftarkan terlebih dahulu dengan menuliskan `int x` di luar blok program utama.
- Pada blok program utama, `x` diisi dengan nilai 12, lalu perintah `printf` dieksekusi.



Sekilas Tentang printf

- Untuk pencetakan, digunakan perintah berikut:

```
printf("Nilai = %d\n", x);
```
- Untuk mencetak nilai dari variabel, diperlukan simbol sementara yang akan digantikan dengan nilai variabel.
- Simbol sementara untuk variabel bertipe bilangan bulat seperti x adalah "%d".
- Variabel-variabel untuk menggantikan simbol sementara perlu dituliskan sesudah pola cetakan.



Contoh Program: assign2.cpp

- Berikut adalah contoh program yang melibatkan beberapa variabel.
-

```
#include <stdio>
```

```
int x;
```

```
int y;
```

```
int main() {
```

```
    x = 12;
```

```
    y = 123456;
```

```
    printf("Nilai x = %d\n", x);
```

```
    printf("Nilai y = %d\n", y);
```

```
    x = 15;
```

```
    printf("Sekarang nilai x = %d\n", x);
```

```
}
```



Penjelasan Program: assign2.cpp

- Keluaran yang dihasilkan dari program itu adalah:

```
Nilai x = 12  
Nilai y = 123456  
Sekarang nilai x = 15
```

- Apa maksud dari kata kunci **int**? Dijelaskan pada bagian selanjutnya.



Bagian 2

Tipe Data Variabel



Tipe Data Variabel

- Setiap variabel pada C++ memiliki **tipe data**.
- Jenis tipe data dasar dari suatu variabel pada:
 - Bilangan bulat.
 - Bilangan riil (bilangan bulat dan pecahan).
 - Karakter (merepresentasikan karakter, seperti 'a', 'b', '3', atau '?').
 - Nilai kebenaran, yaitu benar (**TRUE**) atau salah (**FALSE**).



Tipe Data: Bilangan Bulat

Nama	Jangkauan	Ukuran
short	$-2^{15} .. 2^{15} - 1$	2 byte
unsigned short	$0 .. 2^{16} - 1$	2 byte
int	$-2^{31} .. 2^{31} - 1$	4 byte
unsigned int	$0 .. 2^{32} - 1$	4 byte
long long	$-2^{63} .. 2^{63} - 1$	8 byte
unsigned long long	$0 .. 2^{64} - 1$	8 byte

- C++ menawarkan beberapa tipe data bilangan bulat yang variasinya terletak pada jangkauan nilai yang bisa direpresentasikan dan ukurannya pada memori.
- Dalam memprogram, yang umum digunakan adalah **int** dan **long long**.



Tipe Data: Bilangan Riil

Nama	Jangkauan (magnitudo)	Akurasi	Ukuran
float	$1.5 \times 10^{-45} .. 3.4 \times 10^{38}$	7-8 digit	4 byte
double	$5.0 \times 10^{-324} .. 1.7 \times 10^{308}$	15-16 digit	8 byte

- Biasa disebut dengan *floating point*.
- Tipe data *floating point* bisa merepresentasikan negatif atau positif dari magnitudonya.
- Pada pemrograman, umumnya tipe data *floating point* dihindari karena kurang akurat. Representasi 3 pada *floating point* bisa jadi 2.999999999999999 atau 3.0000000000000001 karena keterbatasan pada struktur penyimpanan bilangan pecahan pada komputer.
- Tipe yang umum digunakan adalah **double**.



Tipe Data: Karakter

- Merupakan tipe data untuk merepresentasikan karakter menurut ASCII (*American Standart Code for Information Interchange*).
- Dalam ASCII, terdapat 128 karakter yang direpresentasikan dengan angka dari 0 sampai 127.
- Misalnya, kode ASCII untuk karakter spasi (' ') adalah 32, huruf 'A' adalah 65, 'B' adalah 66, huruf 'a' adalah 97, dan huruf 'b' adalah 98.
- Pada C++, tipe data ini dinyatakan sebagai **char**, dengan ukuran 1 byte.



Tipe Data: Boolean

- Merupakan tipe data yang menyimpan nilai kebenaran, yaitu hanya **TRUE** atau **FALSE**.
- Tipe data ini akan lebih terasa kebermanfaatannya ketika kita sudah mempelajari struktur percabangan dan **array**.
- Pada C++, kalian dapat menggunakan tipe data **boolean**.



Deklarasi Variabel

- Deklarasi variabel adalah aktivitas mendaftarkan nama-nama dan tipe variabel yang akan digunakan.
- Pada saat dideklarasikan, setiap variabel perlu disertakan tipe datanya.



Deklarasi Variabel (lanj.)

- Pada C++, variabel dapat dideklarasikan di luar atau di dalam blok program.
- Apabila variabel dideklarasikan di luar blok program, artinya variabel tersebut bersifat *global*.
- Tipe data dituliskan sebelum nama variabel, dipisahkan oleh spasi.
Contoh: `"int nilai"` atau `"double rerata"`.
- Beberapa variabel juga bisa dideklarasikan secara bersamaan jika memiliki tipe data yang sama. Contoh: `"double x, y"`.



Contoh Program: typedasar.cpp

- Pahami program berikut ini dan coba jalankan!
-

```
#include <stdio>
```

```
int p1, p2;  
double x, y;
```

```
int main() {  
    p1 = 100;  
    p2 = p1;  
    printf("p1: %d, p2: %d\n", p1, p2);  
  
    x = 3.1418;  
    y = 234.432;  
    printf("x %lf\n", x);  
    printf("y %lf\n", y);  
}
```



Penjelasan Program: `tipedasar.cpp`

- Berikut adalah keluaran dari program `tipedasar.cpp`:

```
p1: 100, p2: 100  
x 3.141800  
y 234.432000
```

- Perhatikan bahwa perintah `p2 = p1` sama artinya dengan `p2 = 100`, karena `p1` sendiri mengacu pada nilai 100.
- Untuk mencetak variabel bertipe `double`, gunakan simbol `"%lf"` (seperti "long float").



Simbol Variabel pada printf

- Sejauh ini, kita mengenal bahwa "%d" digunakan untuk mencetak int, dan "%lf" untuk double.
- Berikut tabel variabel beserta simbolnya:

Variabel	Simbol
short	%d
unsigned short	%u
int	%d
unsigned int	%u
long long	%lld atau %I64d
unsigned long long	%llu atau %I64u
float	%f
double	%lf
char	%c



Simbol Variabel pada printf (lanj.)

- Untuk `boolean`, Anda dapat menggunakan `%d` yang akan mencetak 1 apabila **TRUE** atau 0 apabila **FALSE**.
- Khusus untuk `long long`, simbolnya bergantung pada sistem operasi yang digunakan.
- Untuk sistem operasi berbasis UNIX (Linux dan Mac), gunakan `%lld` dan `%llu`.
- Untuk sistem operasi Windows, gunakan `%I64d` dan `%I64u`.



Tipe Data Komposit: Struct

- Kadang-kadang, kita membutuhkan suatu tipe data yang sifatnya komposit; terdiri dari beberapa data lainnya.
- Contoh kasusnya adalah ketika kita butuh suatu representasi dari titik. Setiap titik pada bidang memiliki dua komponen, yaitu x dan y .



Tipe Data Komposit: Struct (lanj.)

- Memang bisa saja kita mendeklarasi dua variabel, yaitu x dan y . Namun bagaimana jika kita hendak membuat beberapa titik? Apakah kita harus membuat $x_1, y_1, x_2, y_2, \dots$?
Sungguh melelahkan!
- Karena itulah C++ menyajikan suatu tipe data komposit, yaitu `struct`.



Tipe Data Komposit: Struct (lanj.)

- Struct dapat dideklarasikan di luar blok program utama.

```
struct <nama_struct> {  
    <tipe_1> <variabel_1>;  
    <tipe_2> <variabel_2>;  
    ...  
};
```

- Setelah dideklarasikan, sebuah tipe data <nama_struct> sudah bisa digunakan.
- Untuk mengakses nilai dari <variabel 1> dari suatu variabel bertipe struct, gunakan tanda titik (.).



Tipe Data Komposit: Struct (lanj.)

- Sebagai contoh, perhatikan contoh program titik.cpp berikut:

```
#include <stdio>

struct titik {
    int x, y;
};

titik a, b;

int main() {
    a.x = 5;
    a.y = 3;

    b.x = 1;
    b.y = 2;
    printf("%d %d\n", a.x, a.y);
    printf("%d %d\n", b.x, b.y);
}
```



Konsumsi Memori Struct

- Memori yang dibutuhkan bagi sebuah tipe data struct bisa dianggap sama dengan jumlah memori variabel-variabel yang menyusunnya.
- Artinya, struct bernama titik pada contoh titik.cpp mengkonsumsi memori yang sama dengan dua buah longint, yaitu 8 byte.
- Perhitungan ini hanya perkiraan saja, sebab konsumsi memori yang sesungguhnya sulit dilakukan.



Ordinalitas

- Menurut keberurutannya, tipe data dapat dibedakan menjadi tipe data **ordinal** atau **non-ordinal**.
- Suatu tipe data memiliki sifat ordinal jika untuk suatu elemennya, kita bisa mengetahui secara pasti apa elemen sebelum atau selanjutnya. Contoh:
 - Diberikan bilangan bulat 6, kita tahu pasti sebelumnya adalah angka 5 dan sesudahnya adalah angka 7.
 - Diberikan karakter 'y', kita tahu pasti sebelumnya adalah karakter 'x' dan sesudahnya adalah karakter 'z'.
- Dengan demikian, seluruh tipe data bilangan bulat dan karakter adalah tipe data ordinal.



Ordinalitas (lanj.)

- Kebalikannya, suatu tipe data dinyatakan memiliki sifat non-ordinal jika kita tidak bisa menentukan elemen sebelum dan sesudahnya. Contohnya:
 - Diberikan bilangan riil 6, apakah elemen sesudahnya 7, atau 6.1, atau 6.01, atau 6.001, atau 6.00000000001?
- Bilangan *floating point* termasuk dalam tipe data non-ordinal.



Yang Sudah Kita Pelajari...

- Mengenal konsep variabel.
- Mempelajari berbagai tipe data.
- Mempelajari cara deklarasi variabel.
- Mengenal operasi assignment.

