Ekspresi

Tim Olimpiade Komputer Indonesia

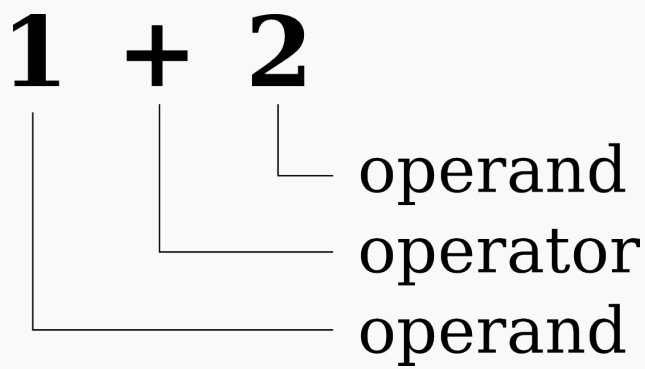
Kilas Balik: Assignment

- Program menjadi kurang bermanfaat jika kita hanya bisa mengisi variabel dengan nilai yang pasti.
- Kadang-kadang dibutuhkan hal yang lebih ekspresif seperti penjumlahan:

```
a = 5;
b = 2;
jumlah = a + b;
```

- Kenyataannya, hal ini dapat diwujudkan pada pemrograman.
- Perintah "a + b" biasa disebut sebagai ekspresi.

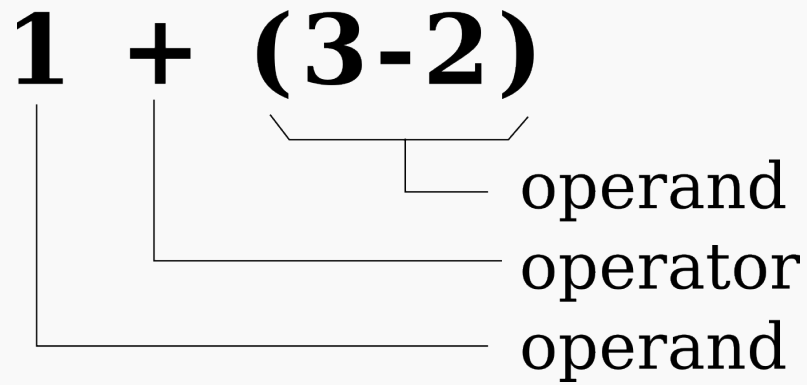
Mengenal Ekspresi



- Ekspresi terdiri dari dua komponen: operator dan operand.
- Operand menyatakan nilai yang akan dioperasikan, misalnya bilangan atau suatu ekspresi lagi.
- Operator menyatakan bagaimana operand akan dioperasikan, apakah ditambah, dikali, atau dibagi?

Mengenal Ekspresi (lanj.)

- Bisa juga dibentuk ekspresi bersarang, yaitu ekspresi yang operand-nya merupakan ekspresi lagi:



Operasi Numerik

- Operasi pada bilangan yang dapat dilakukan adalah penjumlahan (+), pengurangan (-), perkalian (*), pembagian (/), dan modulo (%).
  - Jika kedua operand merupakan bilangan bulat, hasil pengoperasian selalu bilangan bulat juga.
  - Ketika setidaknya salah satu dari operand ada yang bertipe data *floating point*, pengoperasian akan selalu menghasilkan *floating point*.

Operasi Numerik (lanj.)

- Operasi pembagian pada kedua operand berupa bilangan bulat didefinisikan sebagai: membagi, lalu dibulatkan (ke bawah untuk hasil positif, ke atas untuk hasil negatif). Contoh:
  - 7 / 2 = 3
  - 10 / 2 = 5
  - 3 / 5 = 0
  - -5 / 2 = -2
- Operasi pembagian dengan salah satu operand berupa *floating point* akan menghasilkan *floating point* pula. Contoh:
  - 10.0 / 5 = 2.0000000
  - 7 / 2.0 = 3.5000000

Operasi Numerik (lanj.)

- Operasi modulo adalah mengambil sisa bagi dari operand pertama terhadap operand kedua. Contoh:
  - $7 \% 2 = 1$
  - $10 \% 2 = 0$
  - $3 \% 5 = 3$
  - $8 \% 3 = 2$
- Operasi modulo hanya bisa dilakukan apabila kedua operand memiliki tipe data bilangan bulat.

Contoh Program: kuadrat.cpp

- Setelah memahami tentang operasi numerik, coba perhatikan program berikut dan cari tahu apa keluarannya!

```
#include <cstdio>

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);
}
```

Prioritas Pengerjaan

- Seperti pada ilmu matematika, ada juga prioritas pengerjaan pada ekspresi numerik. Tabel berikut menunjukkan prioritasnya:

Prioritas	Operasi
1	*,/,%
2	+,-

- Jika ada beberapa operasi bersebelahan yang memiliki prioritas sama, operasi yang terletak di posisi lebih kiri akan dikerjakan lebih dahulu.

Contoh Program: numerik.cpp

- Kita juga bisa menggunakan tanda kurung untuk mengatur prioritas pengerjaan suatu ekspresi.
- Perhatikan contoh berikut dan coba jalankan programnya:

```
#include <cstdio>

int hasil1, hasil2;

int main() {
  hasil1 = 3+5 / 4;
  hasil2 = (3+5) / 4;
  printf("%d\n", hasil1);
  printf("%d\n", hasil2);
}
```

- Isi dari variabel hasil1 adalah 4, karena operasi "5 div 4" memiliki prioritas yang lebih tinggi untuk dikerjakan, dan menghasilkan nilai 1. Barulah "3 + 1" dilaksanakan.

Operasi Unary

- Pada C++, terdapat pula operasi *unary* numerik.
- Operasi *unary* berarti hanya melibatkan satu operand.
- Misalnya terdapat variabel `x`, operasi *unary* tersedia berupa:
  - `x++`, artinya tambah `x` dengan 1.
  - `x--`, artinya kurangi `x` dengan 1.

Contoh Operasi Unary

Perhatikan dan coba eksekusi program berikut untuk memahami operasi *unary*:

```
#include <cstdio>

int main() {
  int x = 5;

  x++;
  printf("x: %d\n", x);

  x--;
  printf("x: %d\n", x);
}
```

Fungsi Dasar Numerik

Untuk membantu perhitungan, C++ menyediakan fungsi-fungsi pada STL "cmath".

- `round`: membulatkan suatu bilangan pecahan bilangan bulat terdekat (hasilnya tetap bertipe *floating point*). Contoh: `round(1.2)` akan menghasilkan 1.0, sementara `round(1.87)` akan menghasilkan 2.0.
- `sqrt`: mendapatkan akar kuadrat dari suatu bilangan. Contoh: `sqrt(9)` akan menghasilkan 3.00, dan `sqrt(3)` akan menghasilkan 1.73205....

Contoh Program: cmath.cpp

- Perhatikan contoh penggunaan STL cmath berikut:

```
#include <cstdio>
#include <cmath>

int main() {
  printf("%lf\n", sqrt(5));
  printf("%lf\n", round(5.2));
  printf("%lf\n", round(5.6));
}
```

Operasi Relasional

- Kita juga bisa melakukan operasi relasional, yaitu:
  - kurang dari ($<$)
  - lebih dari ($>$)
  - sama dengan ($==$)
  - kurang dari atau sama dengan ($<=$)
  - lebih dari atau sama dengan ($>=$)
  - tidak sama dengan ($!=$)
- Operasi relasional harus melibatkan dua operand (ingat bahwa operand bisa jadi berupa ekspresi lagi), dan menghasilkan sebuah nilai kebenaran.
- Pada C++, nilai kebenaran dinyatakan dengan tipe data **boolean**.

Contoh Program: relasional.cpp

- Perhatikan contoh berikut dan coba jalankan programnya:

```
#include <cstdio>

int main() {
  printf("%d\n", 2 > 1);
  printf("%d\n", 2 < 1);
  printf("%d\n", 2 == 1);
  printf("%d\n", 2 >= 1);
  printf("%d\n", 1 == 1);
  printf("%d\n", 1 != 1);
  printf("%d\n", 1 != 2);
}
```

Operasi Relasional pada Floating Point

- Karena komputer tidak dapat secara sempurna menyimpan nilai *floating point*, Anda perlu hati-hati saat membandingkan dua bilangan riil.
- Ekspresi berikut mungkin saja bernilai **FALSE**:

```
(0.1 + 0.2) == 0.3
```

- Sebab 0.1 + 0.2 bisa saja bernilai **0.30000000000000001**

Operasi Relasional pada Floating Point (lanj.)

- Untuk memeriksa kesamaan antara dua nilai *floating point*, biasanya dilibatkan suatu nilai toleransi.
- Misalnya, kedua nilai dianggap sama apabila selisih mereka kurang dari 10^{-8}.

Operasi Relasional (lanj.)

- Operasi relasional dapat dilakukan pada setiap tipe data ordinal, sehingga bisa juga diterapkan pada **char**.
- Perbandingan karakter dilakukan dengan membandingkan kode ASCII mereka, sehingga menjadi seperti membandingkan angka biasa.
- Contoh:
  - 'a' $<$ 'b' akan bernilai **TRUE**
  - 'a' $>$ 'z' bernilai **FALSE**
  - 'A' $<$ 'a' akan bernilai **TRUE**

Operasi Relasional (string)

- Lebih jauh lagi, **string** sebenarnya merupakan untaian **char**. Operasi relasional juga bisa diterapkan pada **string** (meskipun **string** bukan tipe data ordinal).
- C++ akan membandingkan karakter demi karakter dari kiri ke kanan. Begitu ditemukan ada perbedaan karakter, lebih kecil atau tidaknya suatu string ditentukan oleh karakter tersebut.
  - Contohnya, "aa" $<$ "ab" akan bernilai **TRUE**.
- Jika sampai salah satu string habis dan tidak ditemukan ada perbedaan karakter, maka stirng yang lebih pendek dianggap lebih kecil.
  - Contohnya "a" $<$ "aa" bernilai **TRUE**.

Contoh Program: relasional2.cpp

- Perhatikan contoh berikut dan coba jalankan programnya:

```
#include <cstdio>

int main() {
  printf("%d\n", 'a' > 'A');
  printf("%d\n", 'a' < 'A');
  printf("%d\n", 'a' >= 'A');
  printf("%d\n", 'a' == 'A');

  printf("%d\n", "a" < "aa");
  printf("%d\n", "abcb" > "abca");
  printf("%d\n", "abc" == "abc");
  printf("%d\n", "abc" <= "abc");
}
```

Operasi Boolean

- Operasi **boolean** merupakan operasi yang hanya melibatkan nilai-nilai kebenaran. Terdiri atas: **not** (!), **and** (&&), **or** (||), **xor** ($\hat{}$).
- Operasi-operasi ini sesuai dengan sebuah cabang ilmu matematika yang bernama "aljabar boolean".
- Operasi not merupakan operasi *unary*. Gunanya untuk membalik nilai kebenaran.
- Tabel berikut menunjukkan efek dari penggunaan **not**, yang cara penulisannya dengan tanda seru (!) sebelum variabelnya.

a	!a
TRUE	FALSE
FALSE	TRUE

Operasi Boolean (lanj.)

- Operasi **boolean** yang lainnya merupakan operasi *binary*, yang artinya melibatkan dua operand.
- Tabel berikut menunjukkan efek dari penggunaan operator-operator tersebut:

a	b	a && b	a || b	a ^ b
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE

Operasi Boolean (lanj.)

- Prioritas pengerjaan dari operator **boolean** secara berurutan adalah: **not**, **and**, **or**, **xor**.
- Tanda kurung juga bisa digunakan untuk menentukan operasi mana yang perlu dijalankan terlebih dahulu. Bahkan sangat disarankan untuk selalu menggunakan tanda kurung untuk kejelasan.

Contoh Program: relasional3.cpp

- Perhatikan contoh berikut dan coba jalankan programnya:

```
#include <cstdio>

int main() {
  printf("%d\n", 2 > 1);
  printf("%d\n", !(2 > 1));
  printf("%d\n", (2 > 1) && (3 > 1));
  printf("%d\n", ((2 > 1) || (3 < 1)) && (1 == 1));
  printf("%d\n", (1 != 1) ^ !(1 != 1));
}
```

- Perhatikan bahwa tanda kurung diperlukan dalam ekspresi "not $(2 > 1)$". Dengan tanda kurung, "$2 > 1$" akan dievaluasi terlebih dahulu, menghasilkan nilai **boolean**. Barulah operator **not** bisa mengolah nilai **boolean** tersebut.

Selanjutnya...

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