Percabangan

Tim Olimpiade Komputer Indonesia

Pendahuluan

Melalui dokumen ini, kalian akan:

- Mengenal percabangan.
- Analisa kasus dan mengimplementasikannya pada C++.

Motivasi

- Bebek-bebek Pak Dengklek sedang belajar tentang membedakan bilangan positif, nol, atau negatif.
- Karena bebek-bebek kebingungan, mereka memberikan kalian sebuah bilangan dan meminta kalian menentukan apakah bilangan itu positif atau bukan positif!
- Jika positif, cetak "positif". Jika tidak, jangan cetak apa-apa.

Motivasi (lanj.)

- Sebuah bilangan dinyatakan positif apabila bilangan tersebut lebih dari nol.
- Dengan begitu, kita memerlukan suatu struktur yang memungkinkan "jika bilangan itu lebih dari 0, maka cetak positif".
- Pada C++, hal ini bisa diwujudkan dengan struktur kondisional **if**.

Struktur "if ... then ..."

- Struktur dari penulisan "if ... then ..." adalah:

```
if (<kondisi>) {
  <perintah 1>;
  <perintah 2>;
  ...
}
```

- Dengan <kondisi> adalah suatu boolean.
- Jika nilai <kondisi> adalah **TRUE**, seluruh perintah yang ada di antara blok "{" dan "}" akan dilaksanakan.
- Jika **FALSE**, seluruh perintah yang ada di antara blok "{" dan "}" akan dilewati.

Blok "{ ... }"

- Struktur yang sebenarnya dari penulisan "if ... then ..." adalah:

```
if (<kondisi>)
  <perintah>;
```

- Jika nilai <kondisi> adalah **TRUE**, <perintah> akan dilaksanakan.
- Jika nilai <kondisi> adalah **FALSE**, <perintah> tidak dilaksanakan.
- Lalu di mana bedanya?

Blok "{ ... }" (lanj.)

- Setelah kondisi dari if, sebenarnya hanya satu perintah yang dapat dieksekusi jika <kondisi> bernilai **TRUE**.
- Perhatikan contoh berikut:

```
if (nilai == 10)
   printf("masuk\n");
   printf("lagi\n");
```

Blok "{ ... }" (lanj.)

- Meskipun perintah `printf("masuk\n")` hanya dieksekusi ketika nilai sama dengan 10, `printf("lagi\n")` akan selalu dilaksanakan tanpa peduli isi variabel nilai.
- Blok "{ ... }", akan berperan sebagai "pembungkus" beberapa perintah menjadi "satu" perintah, sehingga berapapun perintah di dalam blok tersebut, akan dilihat oleh C++ sebagai "satu" perintah.

Blok "{ ... }" (lanj.)

- Menulis "{ ... }" setiap sesudah if merupakan kebiasaan yang bagus, meskipun isi dari if itu hanya satu perintah.
- Dengan cara ini, ketika ada tambahan perintah yang perlu dimasukkan ke dalam if, kalian tidak perlu menuliskan lagi "{ ... }".
- Konsisten dengan selalu menulis "{ ... }" juga menjaga program tetap rapi.

Contoh Program: kondisi.cpp

- Ketikkan dan jalankan program berikut:

```
#include <cstdio>

int main() {
  int x;
  scanf("%d", &x);

  if (x > 0) {
    printf("positif\n");
  }
}
```

- Perhatikan bahwa ekspresi "x > 0" akan merupakan operasi relasional yang menghasilkan nilai boolean. Sehingga tepat untuk digunakan pada if.
- Bagaimana jika ingin dibuat jika bilangan itu bukan positif, cetak "non-positif"?

Struktur "if ... then ... else ..."

- Kita juga bisa membuat percabangan jika nilai pada <kondisi> adalah **FALSE**, yaitu dengan kata kunci **else**.
- Struktur dari penulisan "if ... then ... else ..." adalah:

```
if (<kondisi>) {
  <perintah 1>;
  <perintah 2>;
  ...
} else {
  <perintah a>;
  <perintah b>;
  ...
}
```

- Jika nilai <kondisi> adalah **TRUE**, <perintah 1>, <perintah 2>, ..., akan dilaksanakan.
- Jika **FALSE**, <perintah a>, <perintah b>, ..., akan dilaksanakan.

Contoh Program: kondisi2.cpp

- Dengan "if ... then ... else ...", kita bisa memodifikasi kondisi.cpp menjadi kondisi2.cpp:

```
#include <cstdio>

int main() {
  int x;
  scanf("%d", &x);

  if (x > 0) {
    printf("positif\n");
  } else {
    printf("non-positif\n");
  }
}
```

Persoalan Sebenarnya

Ketika bebek-bebek memberikan kalian sebuah bilangan, sebut saja **x**, mereka ingin tahu:

- Jika **x** positif, cetak "positif".
- Jika **x** sama dengan nol, cetak "nol".
- Jika **x** negatif, cetak "negatif".

Pada kasus ini, diperlukan struktur if yang lebih dari dua cabang!

Struktur "if ... then ... else if ..."

- C++ menyediakan struktur yang memungkinkan kita memilah-milah untuk cabang yang lebih dari dua, yaitu dengan struktur "if ... then ... else if ...".
- Struktur dari penulisan "if ... then ... else if ..." adalah:

```
if (<kondisi 1>) {
  <perintah 1>;
  <perintah 2>;
  ...
} else if (<kondisi 2>) {
  <perintah a>;
  <perintah b>;
  ...
} else if (<kondisi 3>) {
  ...
}
```

Struktur "if ... then ... else if ..." (lanj.)

- Jika nilai <kondisi 1> **TRUE**, <perintah 1>, <perintah 2>, ..., akan dilaksanakan.
- Jika nilai <kondisi 1> **FALSE**, diperiksa apakah <kondisi 2> bernilai **TRUE**. Jika ya, <perintah a>, <perintah b>, ..., akan dilaksanakan.
- Jika nilai <kondisi 2> **FALSE**, diperiksa apakah <kondisi 3> bernilai **TRUE**. Hal ini akan terus diulang sampai seluruh percabangan habis.
- Kalian juga bisa mengakhiri struktur ini dengan "else ...", yaitu ketika seluruh kondisi yang diberikan tidak terpenuhi, maka perintah-perintah di bawah **else** ini yang akan dilaksanakan.

Contoh Program: kondisi3.cpp

- Dengan "if ... then ... else if ...", kita bisa memodifikasi kondisi2.cpp menjadi kondisi3.cpp:

```
#include <cstdio>

int main() {
  int x;
  scanf("%d", &x);

  if (x > 0) {
    printf("positif\n");
  } else if (x == 0) {
    printf("nol\n");
  } else if (x < 0) {
    printf("negatif\n");
  }
}
```

Contoh Program: kondisi4.cpp

- Pada kondisi3.cpp, sebenarnya "else if ..." yang terakhir tidak diperlukan.
- Ketika suatu bilangan bukan positif dan bukan nol, sudah pasti bilangan itu negatif. Sehingga bisa didapatkan kondisi4.cpp:

```
#include <cstdio>

int main() {
  int x;
  scanf("%d", &x);

  if (x > 0) {
    printf("positif\n");
  } else if (x == 0) {
    printf("nol\n");
  } else {
    printf("negatif\n");
  }
}
```

Kombinasi dengan Ekspresi Boolean

- Kalian juga bisa menggabungkan struktur if dengan ekspresi **boolean**:

```
if ((x > 0) && (x % 2 == 1)) {
  printf("positif dan ganjil\n");

} else if ((x > 0) && (x % 2 == 0)) {
  printf("positif dan genap\n");

} else if ((x < 0) && (x % 2 == 1)) {
  printf("negatif dan ganjil\n");

} else if ((x < 0) && (x % 2 == 0)) {
  ...
```

If Bersarang

- Solusi yang lebih rapi dicapai dengan menggunakan if secara bersarang:

```
if (x > 0) {
  if (x % 2 == 1) {
    printf("positif dan ganjil\n");
  } else {
    printf("positif dan genap\n");
  }
} else if (x < 0) {
  ...
```

Selanjutnya...

- Ke bagian yang lebih menarik lagi, yaitu perulangan!
- Pastikan kalian menguasai materi percabangan terlebih dahulu.