PART 38

PERCABANGAN KONDISI IF BERSARANG DALAM PASCAL

KONSEP DASAR KONDISI IF BERSARANG (NESTED IF)

Secara sederhana, **IF bersarang** atau **nested IF** adalah penggunaan struktur **IF di dalam IF**. Kondisi seperti ini sering digunakan untuk kode program yang sudah cukup kompleks.

Terdapat banyak variasi dari **nested IF**, tergantung kode program yang ingin kita rancang. Salah satunya adalah sebagai berikut:

```
1   IF (kondisi 1) THEN
2     begin
3       (kode program 1)
4       IF (kondisi 1.1) THEN
5         begin
6           (kode program 1.1)
7         end;
8     end
9   ELSE
10    begin
11      (kode program 2)
12    end;
```

CONTOH KODE PROGRAM NESTED IF DALAM PASCAL

Cukup dengan teori seputar **nested IF** atau **IF bersarang**. Mari kita masuk ke contoh program. Saya ingin membuat sebuah program yang meminta input angka (integer), kemudian menginformasikan apakah angka itu angka genap atau ganjil, dan apakah angka itu besar atau kecil dari 10.

Dapatkah anda merancang kode programnya? Tentunya dengan menggunakan konsep *nested IF*. Disini terdapat 2 kondisi dengan 4 kemungkinan:

1. Angka genap dan besar dari 10
2. Angka genap dan kecil dari 10
3. Angka ganjil dan besar dari 10
4. Angka ganjil dan kecil dari 10

```pascal
program struktur_if_then_else_nested;
uses crt;
var
  angka:integer;
begin
  clrscr;
  write('Masukkan sebuah angka: ');
  readln(angka);
  if (angka mod 2 = 0) then
    begin
      write('Angka yang anda masukkan merupakan bilangan genap ');
      if (angka > 10) then
        begin
          writeln('dan besar dari 10');
        end
      else
        begin
          writeln('dan kecil dari 10');
        end;
    end
  else
    begin
      write('Angka yang anda masukkan merupakan bilangan ganjil ');
     if (angka > 10) then
        begin
          writeln('dan besar dari 10');
        end
      else
        begin
          writeln('dan kecil dari 10');
        end;
    end;
  readln;
end.
```


program struktur_if_then_else_nested;
uses crt;
var
angka:integer;
begin
clrscr;
write('Masukkan sebuah angka: ');
readln(angka);
if (angka mod 2 = 0) then
begin
write('Angka yang anda masukkan merupakan bilangan genap ');
if (angka > 10) then
begin
writeln('dan besar dari 10');
end
else
begin
writeln('dan kecil dari 10');
end;
end
else
begin
write('Angka yang anda masukkan merupakan bilangan ganjil ');
if (angka > 10) then
begin
writeln('dan besar dari 10');
end
else
begin
writeln('dan kecil dari 10');
end;
end;
readln;
end.

Silahkan pelajari sebentar kode program diatas. Jika anda sudah mengikuti seluruh tutorial pasca di duniailkom, saya yakin bisa membaca maksud kode program ini (walaupun agak panjang).

Kemampuan logika juga akan berperan. Sebagai contoh, saya tidak perlu memeriksa apakah angka tersebut kecil dari 10, atau apakah angka itu tidak habis dibagi 2. Kondisi ini cukup dibuat menggunakan **ELSE**. Jika sebuah angka tidak lebih besar dari 10, pastinya angka itu kecil dari 10 (tidak perlu diperiksa). Jika angka tidak habis dibagi 2, berarti angka tersebut ganjil.

Saya juga menggunakan perintah **write** dan **writeln** secara bergantian, agar hasil akhir teks menjadi 1 baris.

Silahkan anda input berbagai angka, dan test apakah kode program tersebut sudah benar atau ada yang kurang.

```
1  program struktur_if_then_else_nested;
2  uses crt;
3  var
4    angka:integer;
5  begin
6    clrscr;
7    write('Masukkan sebuah angka: ');
8    readln(angka);
9    if (angka mod 2 = 0) and (angka > 10) then
10     begin
11       writeln('Angka yang anda masukkan merupakan bilangan genap dan besar d
12     end
13   else
14   if (angka mod 2 = 0) and (angka < 10) then
15     begin
16       writeln('Angka yang anda masukkan merupakan bilangan genap dan kecil d
17     end
18   else
19   if (angka mod 2 <> 0) and (angka > 10) then
20     begin
21       writeln('Angka yang anda masukkan merupakan bilangan ganjil dan besar
22     end
23   else
24   if (angka mod 2 <> 0) and (angka < 10) then
25     begin
26       writeln('Angka yang anda masukkan merupakan bilangan ganjil dan kecil
27     end;
28   readln;
29 end.
```

Yup, kali ini tanpa nested IF dan hasil yang didapat sama persis dengan kode program kita sebelumnya. Bagi kebanyakan orang, kode ini juga lebih mudah dibaca, karena kita langsung bisa melihat kondisi apa saja yang akan diperiksa.

Namun dari sudut pandang performa, **nested IF** lebih efisien daripada ini. Dalam situasi terburuk, variabel angka hanya diperiksa sebanyak 2 kali jika menggunakan **nested IF**. Misalkan saya menginput **angka:= 7**, pertama kali akan diperiksa kondisi: **if (angka mod 2 = 0)**? Tidak, program langsung menjalankan bagian **ELSE**, dan masuk ke kondisi kedua: **if (angka > 10)**? Tidak, jalankan **ELSE**.

Sedangkan yang tanpa nested IF, variabel angka akan diperiksa lebih dari 4 kali. Pertama **if (angka mod 2 = 0)?** Tidak. Program lanjut ke bagian ELSE IF kedua: **if (angka mod 2 = 0)?** Juga tidak, lanjut ke ELSE IF ketiga: **if (angka mod 2 <> 0)?** Betul. Kali ini program akan masuk ke kondisi **(angka > 10)?** Tidak. Sehingga akan dijalankan ELSE terakhir: **if (angka mod 2 <> 0) and (angka < 10)?** Benar.

Untuk kode program yang sederhana seperti diatas, efeknya tidak akan terasa (karena dieksekusi dengan sangat cepat). Beberapa programmer juga tidak keberatan mengorbankan sedikit performa agar kode program mudah dibaca daripada menggunakan **nested IF**.

HATI-HATI DENGAN KESALAHAN LOGIKA!

- Dalam pembuatan kode program, kita sering melakukan kesalahan. Misalnya lupa menambahkan penutup titik koma ‘ ; ‘ diakhir setiap perintah atau salah memberikan tipe data ke sebuah variabel.
- Kesalahan seperti ini disebut sebagai **syntax error**. Meskipun sering terjadi, error seperti ini gampang ditemukan. **Compiler Pascal** (aplikasi yang menerjemahkan kode program yang kita ketik), akan langsung menampilkan pesan error ini sebagai **syntax error**.
- Jenis error yang kedua adalah kesalahan logika (**logic error**). Berbeda dengan *syntax error*, **logic error** sangat susah ditemukan. Error jenis ini tidak akan “diprotes” oleh compiler pascal, karena penulisan kode programnya memang sudah benar. Kesalahan ini disebabkan dari programmernya sendiri yang kurang hati-hati.
- Sebagai contoh, 2 kode program yang saya tulis diatas **mengandung kesalahan logika yang cukup fatal.** Dapatkah anda menemukannya? Kode programnya sendiri memang tidak mengandung *syntax error*, soalnya kalau ada, program tentu tidak akan bisa jalan.