Penunjang Pemrograman Dasar

Tim Olimpiade Komputer Indonesia

Pendahuluan

Dokumen ini berisi tambahan pengetahuan yang dapat menunjang pemrograman dasar kalian.

Melalui dokumen ini, kalian akan:

- Mengenal komentar.
- Memahami pesan kesalahan.
- Memahami I/O *redirection*.

Bagian 1

Komentar

Mengenal Komentar

- Program yang pendek seperti kuadrat.pas atau jumlah.pas yang sebelumnya telah kita jumpai memang sederhana dan mudah dipahami.
- Ketika program sudah mulai panjang dan kompleks, memahami alur kerja suatu program menjadi lebih sulit.
- Salah satu cara untuk membantu memahami alur kerja program adalah dengan menulis komentar.

Komentar

- Merupakan bagian dari program yang diabaikan oleh *compiler*.
- Kita bisa menuliskan apapun di dalam komentar. Misalnya: apa yang dilakukan oleh suatu bagian program atau catatan tertentu.
- Pada Pascal, komentar dapat dituliskan dalam dua gaya:
  - Satu baris, dituliskan dengan awalan dua *slash*

```
// ini adalah komentar, hanya bisa sebaris
// jika perlu baris baru, tambahkan // lagi
```

  - Beberapa baris, dituliskan dengan mengawali dan mengakhiri komentar dengan kurawal ({}) atau kurung bintang ((**)).

```
{ ini adalah komentar, yang memungkinkan
  ditulis dalam beberapa baris }
(* ini juga komentar,
   ditulis dengan cara lain *)
```

Contoh Program: kuadrat3.pas

- Perhatikan program berikut:

```
var
  a, b, c, x, hasil: longint;
begin
  // inisialisasi
  a := 1;
  b := 3;
  c := -2;

  // baca nilai x
  readln(x);

  // hitung hasil fungsi
  hasil := a*sqr(x) + b*x + c;

  // cetak
  writeln('ax^2 + bx + c = ', hasil);
end.
```

Penjelasan Program: kuadrat3.pas

- Program kuadrat3.pas menjadi lebih deskriptif ketika kita menuliskan komentar.
- Ketika program yang dibuat sudah mulai panjang, komentar menjadi efektif untuk membantu kalian "mengingat kembali" apa yang telah diketikkan sebelumnya.
- Komentar juga berguna ketika program kalian akan dibaca oleh orang lain, sehingga orang lain bisa memahaminya dengan lebih mudah.
- Gunakan komentar secukupnya, jangan terlalu berlebihan juga.

Bagian 2

Pesan Kesalahan (Error)

Dua Jenis Error

Compilation Error

Kesalahan yang terjadi ketika program dikompilasi.
Contoh: terdapat kesalahan dalam pengetikan nama variabel, kurang tanda titik koma (;), atau salah penggunaan tipe data.

Runtime Error

Kesalahan yang terjadi ketika program dieksekusi.
Contoh: saat program dieksekusi, tiba-tiba ada operasi pembagian dengan 0.

- Mampu memahami pesan kesalahan yang disampaikan dapat membantu kita memperbaiki program secara lebih efisien.

Compilation Error

- Pada kompilator Free Pascal, pesan kesalahan saat kompilasi biasanya disampaikan dengan format:

```
<nama berkas>(<nomor baris>,<nomor kolom>) Error:
    <jenis error>
```

Contoh:

```
tes.pas(4,13) Error: Identifier not found "nilai"
```

- Artinya pada berkas tes.pas, baris 4, kolom 13, terdapat kesalahan berupa: sebuah *identifier* bernama "nilai" tidak ditemukan. Untuk memperbaikinya, "nilai" harus dideklarasikan terlebih dahulu.

Compilation Error (lanj.)

- Ketika suatu program memiliki *compilation error*, kompilasi akan dibatalkan dan program tidak bisa dikompilasi sampai kesalahannya diperbaiki.

Compilation Warning

- Selain *error*, kompilator Free Pascal juga memberikan beberapa peringatan (*warning*).
- Tidak seperti *error*, *warning* tidak membuat program batal dikompilasi. *Warning* hanya seperti peringatan bahwa ada beberapa bagian dari program kalian yang mungkin bermasalah.

Compilation Warning (lanj.)

- Contoh:

```
tes.pas(5,19) Warning: Variable "saldo" does not seem
    to be initialized
```

- Artinya pada berkas tes.pas, baris 5, kolom 19, terdapat peringatan berupa: sebuah variabel bernama "saldo" belum diinisialisasi (belum diberi nilai awal). Untuk memperbaikinya, "saldo" dapat diinisialisasi terlebih dahulu, misalnya dengan 0 (jika "saldo" memiliki tipe data numerik).

Runtime Error

- Ketika program sudah berhasil dikompilasi, belum tentu program luput dari *error* ketika dieksekusi.
- Program dapat mengalami *error* ketika sedang dieksekusi karena berbagai hal:
  - Melakukan pembagian dengan angka 0.
  - Mengakses memori di luar yang telah dialokasikan.
  - Mengalami *stack overflow*.
- Sebagian besar dari istilah dan masalah yang dijelaskan di atas mungkin kalian hadapi ketika sudah mempelajari tentang array dan rekursi.

Error Code

- Setiap *runtime error* diasosiasikan dengan sebuah kode, biasa disebut dengan *error code*. Contoh:
  - 200 Division by zero
  - 201 Range check error
  - 202 Stack overflow error
  - 204 Invalid pointer operation
  - 205 Floating point overflow
  - 206 Floating point underflow
- Dari *error code* dan nama *error* yang muncul, kalian bisa mendiagnosa masalah pada program dan memperbaikinya.

Bagian 3

IO Redirection

IO Redirection

- Penjelasan tentang saluran input dan output sempat dijelaskan pada Sesi 2. Kali ini, kita akan memperdalamnya.
- Pada contoh yang lalu, kita sempat melakukan hal ini:

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

- Ada dua hal yang dilakukan di sini:
  - Memberikan STDIN kepada program jumlah yang akan dieksekusi dengan input.txt. Hal ini dilakukan dengan operator $<$.
  - Memberikan STDOUT kepada program jumlah yang akan dieksekusi dengan output.txt. Hal ini dilakukan dengan operator $>$.

IO Redirection (lanj.)

- Kita bisa melakukan hanya salah satu dari keduanya. Misalnya jika kita melakukan:

```
jumlah < input.txt
```

- Artinya program jumlah akan dijalankan dengan STDIN dari berkas input.txt, dan STDOUT ke layar.
- Hal ini akan membantu dalam mengurangi pengetikan berkas masukan terus menerus secara manual.
- Demikian pula dengan:

```
jumlah > output.txt
```

- Artinya program jumlah akan dijalankan dengan STDIN dari layar (kalian dapat mengetikkannya), dan STDOUT ke berkas output.txt.

Selanjutnya...

- Memasuki bagian yang menarik, yaitu struktur percabangan.