CARA PENANGANAN KESALAHAN

Teknik Kompilasi

Dosen: Utami Dewi W.,S.Kom

KESALAHAN PROGRAM

- Sebuah kompilator akan sering menemui program yang mengandung kesalahan.
- Oleh karena itu, kompilator harus memiliki strategi yang dapat dilakukan untuk menangani kesalahan-kesalahan tersebut.

JENIS KESALAHAN PROGRAM (1)

1. KESALAHAN LEKSIKAL :

- Yaitu kesalahan dalam mengeja keyword
- Contoh :

  THEN ditulis sebagai TEN / THN

JENIS KESALAHAN PROGRAM (2)

2. KESALAHAN SINTAKS :

- Yaitu kesalahan dalam penulisan sintaks, misalnya pada operasi aritmatika dengan jumlah parenthesis (kurung) yang tidak pas.
- Contoh :

  A : = X + (B*(C+D)

JENIS KESALAHAN PROGRAM (3)

3. KESALAHAN SEMANTIK :

- Kesalahan dalam penentuan tipe data
- Contoh :

  VAR siswa : integer

  siswa := ‘yanuar’

- Kesalahan karena belum mendefinisikan variabel
- Contoh :

  B := B + 1;

  Variabel B belum didefinisikan

PENANGANAN KESALAHAN (1)

- Langkah-langkah
  1. Mendeteksi kesalahan
  2. Melaporkan kesalahan
  3. Tindak lanjut pemulihan/perbaikan

PENANGANAN KESALAHAN (2)

- Sebuah kompilator yang menemukan kesalahan akan melakukan pelaporan, yang biasanya meliputi :
  1. Kode kesalahan
  2. Pesan kesalahan dalam bahasa manual
  3. Nama & atribut identifier
  4. Tipe-tipe yang terkait dengan *type checking*

  Misal : Error 162 Jumlah : unknown identifier

  Artinya : kode kesalahan = 162, pesan kesalahan = *unknown identifier,* nama identifier = jumlah

REAKSI KOMPILATOR (1)

- Reaksi-reaksi yang **tidak dapat diterima**
  1. Kompilator crash : berhenti atau hang
  2. Looping : kompilator masih berjalan, tapi tidak pernah berhenti karena looping yang tak berhingga.
  3. Menghasilkan program objek yang salah : kompilator melanjutkan proses sampai selesai tapi program objek yang dihasilkan salah ketika saat dieksekusi.

REAKSI KOMPILATOR (2)

- Reaksi yang **benar, tetapi kurang dapat diterima** dan kurang bermanfaat
  1. Kompilator menemukan kesalahan pertama, melaporkannya, lalu berhenti (halt).
  2. Kompilator hanya mampu mendeteksi dan melaporkan kesalahan satu kali kompilasi.
  3. Pemrogram akan membuang waktu untuk melakukan pengulangan kompilasi setiap kali terdapat *error*.

REAKSI KOMPILATOR (3)

- Reaksi-reaksi yang **dapat diterima**
  1. Reaksi yang sudah dapat dilakukan, yaitu kompilator melaporkan error, selanjutnya melakukan *recovery* dan *repair*.
  2. Reaksi yang belum dapat dilakukan, yaitu kompilator mengoreksi kesalahan lalu menghasilkan program objek sesuai dengan yang diinginkan pemrogram.

ERROR RECOVERY (1)

- Bertujuan untuk mengembalikan parser ke kondisi stabil agar dapat melanjutkan proses parsing ke posisi selanjutnya.
- Strategi error recovery :
  1. Mecanisme Ad Hoc
  2. Syntax directed recovery
  3. Secondary Error Recovery
  4. Context sensitive recovery.

ERROR RECOVERY (2)

1. MEKANISME AD HOC

- Recovery yang dilakukan tergantung dari pembuat kompilator itu sendiri.
- Tidak terikat pada suatu aturan tertentu.
- Disebut juga dengan istilah *special purpose error recovery.*

ERROR RECOVERY (3)

2. SYNTAX DIRECTED RECOVERY

- Melakukan recovery berdasarkan syntax.
- Misalnya :

```
begin
A := A + 1
B := B + 1;
C := C + 1
end;
```

- Kompilator akan mengenali sebagai (dalam notasi BNF)
- Begin <statement>? <statement>; <statement> end;
- ? Akan diperlakukan sebagai ;

ERROR RECOVERY (4)

3.SECONDARY ERROR RECOVERY

- Berguna untuk melokalisir error, caranya :
  - *Panic mode*

    Maju terus dan mengabaikan teks sampai bertemu delimiter (misal ‘;’). Contoh :

    ```
    IF A = 1
        kondisi := true
    ```

    Pada contoh di atas, kesalahan karena tidak ada instruksi THEN, kompilator akan terus maju sampai bertemu dengan titik koma.
  - Unit deletion

    Menghapus keseluruhan suatu unit sintaktik, (misal : <block>, <exp>, <statement> dsb). Efeknya mirip dengan panic mode tetapi unit deletion memelihara kebenaran syntax dari source program dan mempermudah untuk melakukan error repairing lebih lanjut.

ERROR RECOVERY (5)

4. CONTEXT SENSITIVE RECOVERY

- Berkaitan dengan semantics, missal bila terdapat variabel yang belum dideklarasikan, maka diasumsikan tipenya berdasarkan kemunculannya.
- Contoh

    B := ‘nama’

  B belum dideklarasikan, maka B diasumsikan sebagai string.

ERROR REPAIR (1)

- Bertujuan untuk memodifikasi source program dari kesalahan dan membuatnya valid
- Mekanisme error repair:
  1. Mecanisme Ad Hoc
  2. Syntax directed repair
  3. Context sensitive recovery.
  4. Spelling repair

ERROR REPAIR (2)

1. MEKANISME AD HOC

- Repair yang dilakukan tergantung dari pembuat kompilator itu sendiri.

ERROR REPAIR (3)

2. SYNTAX DIRECTED REPAIR

- Menyisipkan symbol terminal yang dianggap hilang atau membuang terminal penyebab kesalahan.
- Contoh :

```
WHILE A < 1
    I := I + 1
```

- Contoh lain :

```
Procedure increment;
begin
X := X + 1; end; end;
```

ERROR REPAIR (4)

3. CONTEXT SENSITIVE REPAIR

- Perbaikan dilakukan pada kesalahan
- Tipe Identifier, Diatasi dengan membangkitkan identifier *dummy*,

misalnya :

```
VAR A : string
begin
A := 0;
end;
```

Kompilator akan memperbaiki kesalahan dengan membangkitkan identifier baru, misalnya B yang bertipe integer.

- Tipe konstanta, diatasi dengan membangkitkan konstanta baru dengan tipe yang tepat

ERROR REPAIR (5)

4. SPELLING REPAIR

- Memperbaiki kesalahan pengetikan pada identifier.
- Misal :

    WHILLE A=1 DO

    identifier yang salah tersebut akan diperbaiki menjadi WHILE