BAB VI
CARA PENANGANAN KESALAHAN

TUJUAN PRAKTIKUM

1. Mengetahui jenis - jenis kesalahan yang biasa terjadi
2. Memahami cara – cara penanganan kesalahan
3. Mengerti bagaimana cara pemulihan kesalahan

TEORI PENUNJANG

Sebuah kompilator akan sering menemui program yang mengandung kesalahan, maka kompilator harus memiliki strategi apa yang harus dilakukan untuk menangani kesalahan - kesalahan tersebut.

6.1 Jenis – jenis Kesalahan

- Kesalahan Leksikal

  Misalnya kesalahan mengeja *keyword*,

  contoh: then ditulis ten
- Kesalahan Sintaks

  Misalnya pada operasi aritmatika kekurangan jumlah *paranthesis* (kurung).

  contoh : A:=X+(B*(C+D)
- Kesalahan Semantik

  Tipe data yang salah, misal tipe data *integer* digunakan untuk variabel *string*.

  Variabel belum didefinisikan tetapi digunakan dalam operasi.

6.2 Penanganan Kesalahan

- Prosedur penanganan kesalahan terdiri dari :
  - Mendeteksi kesalahan

- Melaporkan kesalahan
- Tindak lanjut perbaikan

■ Pelaporan kesalahan yang dilakukan oleh sebuah kompilator meliputi :

- Kode kesalahan
- Pesan kesalahan dalam bahasa natural
- Nama dan atribut *identifier*
- Tipe – tipe yang terkait bila *type checking*

  Contoh : *Error* 162 *jumlah: unknown identifier*
  - Kode kesalahan = 162
  - Pesan kesalahan = *unknown identifier*
  - Nama identifier = *jumlah*

■ Pada saat kompilator menemukan kesalahan terdapat beberapa tingkatan diantaranya adalah :

a. Reaksi yang tidak dapat diterima (tidak melaporkan *error*)
   - Kompilator *crash* : berhenti atau *hang*
   - *Looping*
   - Kompilator melanjutkan proses sampai selesai tapi program program objek yang dihasilkan salah.

b. Reaksi yang benar tapi kurang dapat diterima dan kurang bermanfaat. Kemampuan kompilator untuk mendeteksi dan melaporkan kesalahan hanya satu kali untuk setiap kali kompilasi.

c. Reaksi yang dapat diterima
   - Reaksi yang sudah dapat dilakukan (dewasa ini), yaitu melaporkan kesalahan, dan selanjutnya melakukan:
     - *Recovery* / pemulihan, lalu melanjutkan menemukan kesalahan yang lain bila masih ada.
     - *Repair* / Perbaikan kesalahan, lalu melanjutkan proses translasi dan menghasilkan program objek yang valid

- Reaksi yang belum dapat dilakukan (dewasa ini), yaitu kompilator mengkoreksi kesalahan, lalu menghasilkan program objek sesuai dengan yang diinginkan pemrogram.

6.3 Pemulihan Kesalahan

Tujuannya mengembalikan *parser* ke kondisi stabil (supaya bisa melanjutkan proses *parsing* ke posisi selanjutnya). Strategi yang dilakukan sebagai berikut :

❖ *Mekanisme Ad Hoc*

Recovery yang dilakukan tergantung dari pembuat kompilator sendiri/Spesifik, dan tidak terikat pada suatu aturan tertentu. Cara ini biasa disebut juga *special purpose error recovery*.

❖ *Syntax Directed Recovery*

Melakukan *recovery* berdasarkan *syntax*

Contoh : ada program

```
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 “;”

❖ *Secondary Error Recovery*

Berguna untuk melokalisir kesalahan, caranya :

- *Panic mode*

  Maju terus dan mengabaikan teks sampai bertemu delimeter (misal ‘:’)

  contoh :

```
if A := 1
    Kondisi := true;
```

Teks diatas terjadi kesalahan karena tidak ada instruksi THEN, kompilator akan maju terus sampai bertemu ';'

- *Unit deletion*

  Menghapus keseluruhan suatu unit sintaktik (misal: <block>,<exp>,<statement> dan sebagainya), efeknya sama dengan *panic mode* tetapi *unit deletion* memelihara kebenaran sintaksis dari source program.

❖ *Context Sensitive Recovery*

Berkaitan dengan semantik, misal bila terdapat variabel yang belum dideklarasikan (*undifined variabel*) maka diasumsikan tipenya berdasarkan kemunculannya.

6.4 Error Repair

Bertujuan untuk memodifikasi *source program* dari kesalahan dan membuatnya valid. Mekanisme *error repair* meliputi :

➢ *Mekanisme Ad Hoc*

Tergantung dari pembuat kompilator sendiri

➢ *Syntax Directed Repair*

Menyisipkan simbol terminal yang dianggap hilang atau membuang terminal penyebab kesalahan

Contoh :

```
While a<1
    I:=I+1;
```

Kompilator akan menyisipkan DO karena kurang simbol DO

➢ *Context Sensitive Repair*

Perbaikan dilakukan pada kesalahan :

- Tipe *identifier*. Diatasi dengan membangkitkan *identifier dummy*, misalkan :

```
Var A : string;
begin
```

```
A:=0;
end;
```

- Tipe konstanta

  Diatasi dengan membangkitkan konstanta baru dengan tipe yang tepat.

- *Spelling repair*

  Memperbaiki kesalahan pengetikan pada identifier, misal :

  WHILLE A = 1 DO

  Identifier yang salah tersebut akan diperbaiki menjadi WHILE

LAPORAN PENDAHULUAN

1. Sebutkan dan berikan contoh dari jenis – jenis kesalahan!
2. Sebutkan dan jelaskan macam – macam pemulihan kesalahan!

LAPORAN AKHIR

Buat program untuk mendeteksi dan mengoreksi kesalahan – kesalahan, baik leksikal, sintaks, maupun semantik!