PERTEMUAN 10

PENGURAIAN ATAS BAWAH (TOP DOWN PARSING) ANALISIS SINTAKS (2)

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi pada pertemuan ini, Mahasiswa diharap mampu menjelaskan konsep penguraian dan penyusunan suatu kalimat atau statemen program.

B. URAIAN MATERI

1. Pengurangan Turun Berulang (Recursive Descent Parsing)

Tabel penguraian prakira adalah usatu tabel yang dipakai untuk menampung produksi-produksi dari taa bahasa yang digunakan dalam proses penguraian. Terdapat suatu algoritma untuk membuat tabel penguraian dari suatu tata bahasa, yaitu sebagai berikut:

Misalkan $A \rightarrow \alpha$ adalah suatu produksi dengan α di dalam $FIRST(\alpha)$. Jika masukannya adalah α, maka pengurai akan mengembakan A dengan α. Tetapi apabila $\alpha = \varepsilon$ atau $\alpha => \varepsilon$, maka pengurai akan mengembakan A dengan α apabila simbol masukan terdapat di dalam $FOLLOW(A)$, atau apabila \$ pada masukan yang telah dicapai dan \$ terdapat di dalam $FOLLOW(A)$.

Berikut ini adalah sebuat algoritma pembentukan tabel penguraian prakira. Pertama-tama siapkan dulu tata bahasa yang akan dipakai. Kemudian lakukan langkah-langkah berikut ini:

a. Untuk setiap produksi $A \rightarrow \alpha$, lakukan langkah 2 dan 3 berikut ini.
b. Untuk setiap terminal α di dalam $FIRST(\alpha)$, tambahkan $A \rightarrow \alpha$ kedalam $M[A, \alpha]$.
c. Jika $\$\epsilon FIRST(\alpha)$, tambahkan $A \rightarrow \alpha$ ke $M[A, b]$ untuk setiap terminal b di dalam $FOLLOW(A)$. Jika $\$\epsilon FIRST(\alpha)$ dan $\$\epsilon FOLLOW(A)$, tambahkan $A \rightarrow \alpha$ ke dalam $M[A, \$]$.
d. Nyatakan masukan yang tidak terdefinisi sebagai *error*.

Apabila semua produksi yang dipakai dalam tata bahasa sudah diproses, maka akan diperoleh keluaran dari proses berupa sebuah tabel penguraian M.

Contoh 1.Dengan mempergunakan tata bahasa (3), dan dengan mempergunakan algoritma pembentukan tabel penguraian prakira di atas, maka dapat dibuat tabel penguraian prakira untuk tata bahasa (3) tersebut, yaitu:

a. Dengan mempergunakan aturan nomor 2, produksi $E \rightarrow TE'$ dimasukkan ke $M[E, var]$ dan $M[E, (]$, karena $FIRST(T) = \{var, (\}$.
b. Dengan mempergunakan aturan nomor 2, produksi $E' \rightarrow +TE'$ dimasukkan ke $M[E', +]$ karena $FIRST(E' + ') = \{+\}$.
c. Dengan mempergunakan aturan nomor 3, produksi $E' \rightarrow \varepsilon$, dimasukkan ke $M[E',)]$ dan $M[E', \$]$ karena $FOLLOW(E') = \{), \$\}$.
d. Dengan mempergunakan aturan nomor 2, produksi $T \rightarrow FT'$ dimasukkan ke $M[T, var]$ dan $M[T, (]$, karena $FIRST(f) = \{VAR, (\}$.
e. Dengan mempergunakan aturan nomor 2, produksi $T' \rightarrow {}^{*}FT'$ dimasukkan ke $M[T', *]$ karena $FIRST(' * ') = \{*\}$.
f. Dengan mempergunakan aturan nomor 3, produksi $T' \rightarrow \varepsilon$, dimasukkan ke $M[T', +]$, $M[T',)]$ dan $M[T', \$]$ karena $FOLLOW(T') = \{+,), \$\}$.
g. Dengan mempergunakan aturan nomor 2, produksi $F \rightarrow (E)$ dimasukkan ke $M[F, (]$ karena $FIRST\ ('(') = \{(\}$.
h. Dengan mempergunakan aturan nomor 2, produksi $F \rightarrow var$ dimasukkan ke $M[F, var]$ karena $FIRST\ (var) = \{var\}$.

Sehingga tabel penguraian prakira untuk tata bahasa (3) adalah sebagai berikut:

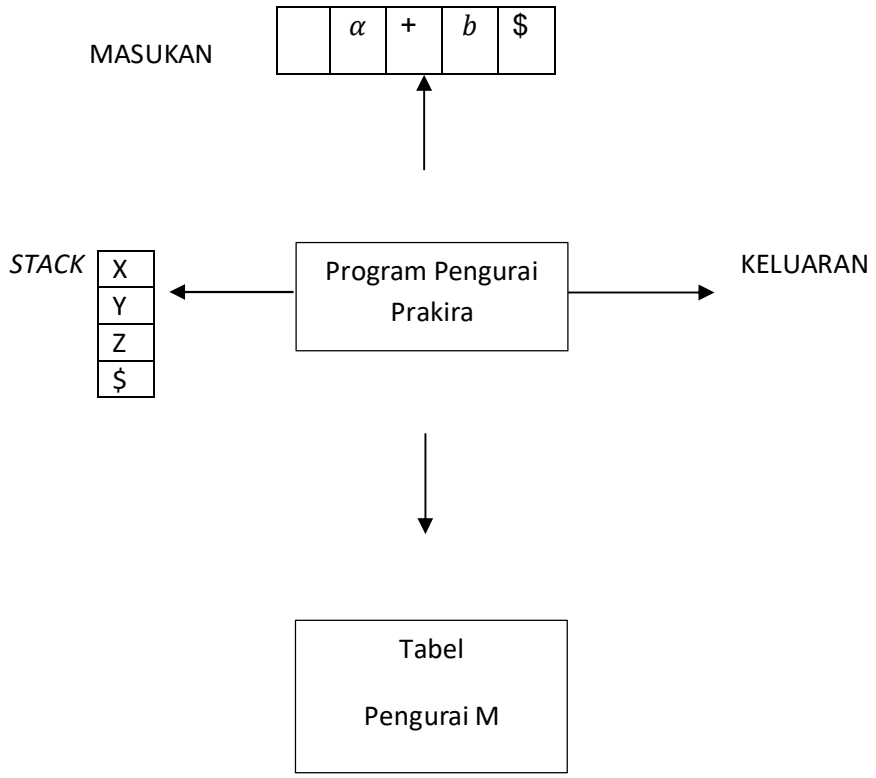
Tabel 10.1 Tabel Penguraian M untuk Tata Bahasa (3)

NON-TERMINAL	SIMBOL MASUKAN					
	Var	+	*	(	)	$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow * FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow var$			$F \rightarrow (E)$		

a. Penguraian Prakira yang *Nonrecursive*

Penguraian prakira dapat dibuat *nonrecursive*, yaitu apabila menggunakan *stack* secara langsung. Persoalan pokok selama proses penguraian adalah menentukan produksi yang mana yang akan dipakai untuk mencari *non-terminal*. Pengurai prakira yang *nonrecursive* yang digambarkan dalam gambar 10.1 mencari produksi yang akan dipakai dalam tabel penguraian.

Pengurai prakira yang dihasilkan dengan tabel ini mempunyai penyangga masukan, *stack*, tabel penguraian, dan keluaran. Penyangga masukan berisi *string* yang akan diurai, diikuti oleh simbol $, yaitu simbol yang digunakan sebagai tanda akhir di sebelah kanan yang menyatakan akhir dari *string* masukan. *Stack* berisi sederetan simbol-simbol dari tata bahasa dengan $ sebagai dasar dari *stack*, yang menyatakan dasar dari *stack* tersebut. Pada awal *stack* akan berisi simbol awal dari tata bahasa dengan $ di bawahnya. Tabel penguraian adalah larik dua dimensi yang dilambangkan dengan $M[A, a]$, A adalah *non-terminal dan sedangkan* a adalah *terminal* (simbol $).



Gambar 10.1 Model Pengurai Prakira yang *Nonrecursive*

Pengurai dikendalikan oleh program yang cara bekerjanya adalah sebagai berikut: Program memperhatikan X sebagai simbol paling atas dari stack, dan simbol masukan α yang sedang dimasukkan. Kedua simbol ini yang akan menentukan aksi dari pengurai. Ada tiga kemungkinan yang muncul:

1) Jika $X = \alpha = \$$, maka pengurai akan berhenti bekerja dan melaporkan bahwa proses penguraian telah selesai.
2) Jika $X = \alpha \neq \$$, maka pengurai akan mem-$pop\ X$ sehingga X keluar dari *stack* dan *pointer* masukan dimajukan menuju simbol masukan yang selanjutnya.
3) Jika X adalah suatu *non-terminal*, maka program akan melihat masukan $M[X, a]$ dari tabel penguraian M. Masukan ini akan berupa produksi X dari suatu tata bahasa atau suatu masukan *error*.

Contoh 1, jika $M[X, a] = \{X \rightarrow UVW\}$, maka pengurai akan mengganti X yang berada di puncak dari *stack* dengan *WVU* dan *U* berada di puncak *stack*. Sebagai keluaran, kita harus mengasumsikan bahwa pengurai hanya mencetak produksi yang dipakai, dan kode yang lainnya dapat dieksekusi

disini. Jika $M[X, a] = error$, maka pengurai akan memanggil rountine pemulihan kesalahan.

Berikut ini adalah algoritma penguraian prakira yang *nonrecursive*. Sebelum proses penguraian, $string\ w$ dan tabel penguraian M dari suatu tatabahasa G harus sudah disediakan. Mula-mula pengurai menyusun $\$S$ pada *stack* dengan S sebagai simbol awal dari G, dan $w\$$ di dalam penyangga masukan. Kemudian dilakukan pengujian pada X dan a sampai diperoleh $X = \$$. Maka jika w terdapat di dalam $L(G)$, akan dihasilkan penurunan paling kiri, tetapi jika tidak akan diberikan indikasi kesalahan.

Contoh 2, Dengan mempergunakan tatabahasa (3), tabel penguraian Tabel 10.1, dan masukan $var + var * var$, pengurai prakira membuat urutan kegiatan sebagai berikut:

1) MASUKAN :

var	+	var	*	var	$

STACK :

E
$

sehingga $X = E$ dan $a = var$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[E, var]$, sehingga diperoleh:

2) KELUARAN : $E \rightarrow TE'$

MASUKAN :

var	+	var	*	var	$

STACK :

T
E'
$

sehingga $X = T$ dan $a = var$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[T, var]$, sehingga diperoleh:

3) KELUARAN: $T \rightarrow FT'$

MASUKAN :

var	+	var	*	var	$

STACK :

F
T
E'
$

sehingga $X = F$ dan $a = var$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[F, var]$, sehingga diperoleh:

4) KELUARAN: $F \rightarrow var$

MASUKAN :

var	+	var	*	var	$

STACK :

var
T'
E'
$

sehingga $X = var$ dan $a = var$. Dengan berdasarkan aturan nomor 2 diperoleh:

5) MASUKAN:

var	+	var	*	var	$

STACK :

T'
E'
$

sehingga $X = T'$ dan $a = +$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[T', +]$, sehingga diperoleh:

6) KELUARAN: $T' \rightarrow \varepsilon$

MASUKAN :

var	+	var	*	var	$

STACK :

E'
$

sehingga $X = E'$ dan $a = +$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[E', +]$, sehingga diperoleh:

7) KELUARAN: $E' \rightarrow +TE'$

MASUKAN :

var	+	var	*	var	$

STACK :

+
T
E'
$

sehingga $X = +$ dan $a = +$. Dengan berdasarkan aturan nomor 2 diperoleh:

8) MASUKAN:

var	+	var	*	var	$

STACK :

T
E'
$

sehingga $X = T$ dan $a = var$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[T, var]$, sehingga diperoleh:

9) KELUARAN: $T \rightarrow FT'$

MASUKAN :

var	+	var	*	var	$

STACK :

F
T'
E'
$

sehingga $X = F$ dan $a = var$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[F, var]$, sehingga diperoleh:

10) KELUARAN: $F \rightarrow var$

MASUKAN :

var	+	var	*	var	$

STACK :

var
T'
E'
$

sehingga $X = var$ dan $a = var$. Dengan berdasarkan aturan nomor 2 diperoleh:

11) MASUKAN:

var	+	var	*	var	$

STACK :

T'
E'
$

sehingga $X = T'$ dan $a = *$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[T', *]$, sehingga diperoleh:

12) KELUARAN: $T' \rightarrow * FT'$

MASUKAN :

var	+	var	*	var	$

STACK :

$*$
F
T'
E'
$

sehingga $X = var$ dan $a = var$. Dengan berdasarkan aturan nomor 2 diperoleh:

13) MASUKAN:

var	+	var	*	var	$

STACK :

F
T'
E'
$

sehingga $X = F$ dan $a = var$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[F, var]$, sehingga diperoleh:

14) KELUARAN: $F \rightarrow var$

MASUKAN :

var	+	var	*	var	$

STACK :

var
T'
E'
$

sehingga $X = var$ dan $a = var$. Dengan berdasarkan aturan nomor 2 diperoleh:

15) MASUKAN:

var	+	var	*	var	$

STACK :

T'
E'
$

sehingga $X = T'$ dan $a = \$$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[T', \$]$, sehingga diperoleh:

16) KELUARAN: $T' \rightarrow \varepsilon$

MASUKAN :

var	+	var	*	var	$

STACK :

E'
$

sehingga $X = E'$ dan $a = \$$. Dengan berdasarkan aturan nomor 3, maka harus melihat tabel $M[E', \$]$, sehingga diperoleh:

17) KELUARAN: $E' \rightarrow \varepsilon$

MASUKAN :

var	+	var	*	var	$

STACK :

$

dengan aturan nomor 1, maka proses penguraian selesai.

2. Tatabahasa LL(1)

Tatabahasa LL(1) adalah tatabahasa yang tabel penguraiannya tidak mengandung masukan-masukan yang mempunyai lebih dari satu definisi. L yang pertama pada LL(1) mengandung arti melihat (*scanning*) masukan dari kiri (*left*) ke kanan, L yang kedua berarti untuk menghasilkan penurunan paling kiri (*leftmost derivation*), dan 1 berarti pemakaian satu simbol masukan yang dilihat pada setiap langkah untuk membuat kegiatan penguraian.

Tatabahasa LL(1) mempunyai beberapa sifat yang penting yaitu tidak ambigu dan rekursif kiri. Tatabahasa G adalah LL(1) jika dan hanya jika untuk setiap $A \rightarrow \alpha|\beta$ dua produksi yang berbeda kondisi berikut ini dipenuhi:

a. $FIRST(\alpha)$ dan $FIRST(\beta)$ saling asing.
b. Paling banyak satu dari α atau β yang dapat menurunkan $empty\ string(\varepsilon)$.
c. Jika $\beta \Rightarrow \varepsilon$, maka α tidak menurunkan *string* yang diawali dengan suatu *terminal* di dalam $FOLLOW(A)$.

Contoh 3, tatabahasa (3) adalah tatabahasa LL(1) karena untuk setiap produksi yang mempunyai dua alternatif ketiga syarata LL(1) dipenuhi, yaitu:

a. Pada produksi $E' \rightarrow +TE'|\varepsilon$, maka $FIRST(+TE') = \{+\}, FIRST(\varepsilon) = \{\varepsilon\}$, dan $\{+\}$ dengan $\{\varepsilon\}$ saling asing. Kemudian pada produksi $T' \rightarrow * FT'|\varepsilon$, maka

$FIRST\ (*\ FT') = \{*\}, FIRST(\varepsilon) = \{\varepsilon\}$, dan $\{+\}$ dengan $\{\varepsilon\}$ saling asing. Berarti syarat a) dipenuhi untuk kedua produksi di atas.

b. Syarat b). juga dipenuhi karena dari kedua produksi tersebut pada masing-masing produksi hanya satu alternatif yang dapat menurunkan ε.

c. Pada produksi $E' \rightarrow +TE'|\varepsilon$, maka $FIRST(+TE') = \{+\}, FOLLOW(E') = \{), \$\}$, dan $\{+\}$ dengan $\{), \$\}$ saling asing. Kemudian pada produksi $T' \rightarrow * FT'|\varepsilon$, maka $FIRST(* FT') = \{*\}, FOLLOW(T') = \{+,), \$\}$, dan $\{*\}$ dengan $\{+,), \$\}$ saling asing. Berarti syarat c dipenuhi untuk kedua produksi di atas.

Karena ketiga syarat dipenuhi maka tatabahasa (3) adalah tatabahasa LL(1).

3. Pembenahan Kesalahan dalam Penguraian Prakira

Kesalahan dalam penguraian prakira akan terdeteksi ketika *terminal* yang berada di puncak dari *stack* tidak sesuai dengan simbol masukan berikutnya, atau ketika *non-terminal A* terdapat di puncak *stack* dengan a sebagai simbol masukan ternyata pada tabel penguraian masukan pada $M[A, a]$ kosong.

Terdapat dua metode pembenahan kesalahan yang dapat dipakai, yaitu metode pembenahan kesalahan model-panik (*panic-mode error recovery*) dan metode pembenahan tingkat frasa (*phrase-level recovery*), yang masing-masing akan diuraikan dalam subbab berikut ini.

a. Pembenahan Kesalahan Model-Panik

Pembenahan kesalahan dengan model-panik didasarkan pada ide untuk melewatkan simbol-simbol pada masukan sampai suatu *token* yang terdapat di dalam himpunan *token* yang benar muncul. Keefektifan metode ini bergantung pada pemilihan himpunan tadi. Himpunan harus dipilih sehingga pengurai akan berbenah dengan cepat dari kesalahan yang mungkin terjadi dalam prakter. Berikut ini adalah beberapa petunjuk yang mungkin bermanfaat:

1) Sebagai langkah awal pilih semua simbol dalam $FOLLOW(A)$ sebagai himpunan *token* tersinkronisasi untuk *non-terminal A*. Jika kita melewatkan token-token sampai suatu elemen dari $FOLLOW(A)$ muncul dan mem-*pop A* dari *stack*, maka kemungkinan besar penguraian akan berlanjut.
2) Jika dengan langkah awal di atas penguraian tetap belum dapat dilaksanakan, maka tambahkan simbol yang terdapat dalam $FIRST(A)$

kedalam himpunan tersinkronisasi untuk *non-terminal A*. Jika langkah kedua ini dilaksanakan, maka penguraian dapat dilanjutkan berdasarkan *A* jika suatu simbol dalam $FIRST(A)$ muncul dalam masukan.

3) Jika suatu *non-terminal* dapat membentuk *empty string*, maka produksi yang menurunkan ε dapat dijadikan sebagai *default*. Hal ini akan menunda pendeteksian beberapa kesalahan tetapi tidak dapat menghindari kesalahan itu. Pendekatan ini mengurangi jumlah *non-terminal* yang harus diperhatikan selama penulisan kesalahan.
4) Jika suatu *terminal* pada puncak *stack* tidak dapat dipasangkan, maka *pop*-lah *terminal* tersebut. Kemudian lanjutkan dengan mengeluarkan suatu pesan bahwa *terminal* tadi dimasukkan dan lanjutkan penguraiannya.

Contoh 4, dengan memakai simbol-simbol dalam $FOLLOW$ dan $FIRST$ sebagai *token* tersinkronisasi, penguraian ekspresi berdasarkan tatabahasa (3) dapat dilakukan dengan baik. Tabel penguraian untuk tatabahasa (3) ditulis kembali dalam tabel 10.2. dengan tambahan masukan *"synch"* untuk menyatakan *token* tersinkronisasi yang diperoleh dari himpunan $FOLLOW$ untuk setiap *non-terminal* yang ada diperoleh contoh berikut:

$$FOLLOW(E) = FOLLOW(E') = \{), \$\};$$

$$FOLLOW(T) = FOLLOW(T') = \{+,), \$\};$$

$$FOLLOW(F) = \{*, +,), \$\}$$

Tabel 10.2 Tabel Penguraian M untuk Tata Bahasa (3)

dengan Tambahan *Token* Pensinkronisasi

NON-TERMINAL	SIMBOL MASUKAN					
	Var	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$	$synch$	$synch$
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$	$synch$		$T \to FT'$	$synch$	$synch$
T'		$T' \to \varepsilon$	$T' \to * FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to var$	$synch$	$synch$	$F \to (E)$	$synch$	$synch$

Tabel 10.2. di atas dipakai seperti berikut:

1) Jika pengurai melihat masukan pada $M[A, a]$ dan menemukan bahwa isinya kosong, maka simbol masukan a dilewatkan.
2) Jika pengurai melihat masukan pada $M[A, a]$ adalah $synch$, *maka non-terminal* yang terdapat di puncak *stack* di-*pop* sebagai usaha untuk melanjutkan penguraian.
3) Jika suatu *token* di puncak *stack* tidak sesuai dengan simbol masukan, maka *token* tersebut di-*pop* dari *stack*, hal ini sesuai dengan aturan nomor (4) di atas.

Apabila digunakan masukan yang salah, yaitu) var * + var, mekanisme penguraian dan pembenahan kesalahan dengan tabel penguraian 10.2., ditunjukkan dalam tabel berikut:

Tabel 10.3 Tabel Penguraian

STACK	MASUKAN	CATATAN
$\$E$	$) \; var * \; + var \; \$$	*Error*, lewati simbol)
$\$E$	$var * + var \; \$$	var berada di dalam $FIRST(E)$
$\$E'T$	$var * + var \; \$$	
$\$E'T'F$	$var * + var \; \$$	$error, M[F, +] = synch$
$\$E'T'var$	$var * + var \; \$$	*F* sudah di-*pop* dari *stack*
$\$E'T'$	$* + var \; \$$	
$\$E'T'F'$	$* + var \; \$$	
$\$E'T'F$	$+ var \; \$$	
$\$E'T'$	$+ var \; \$$	
$\$E'$	$+ var \; \$$	
$\$E'T +$	$+ var \; \$$	
$\$E'T$	$var \; \$$	

$\$E'T'F$	$var\ \$$	
$\$E'T'var$	$var\ \$$	
$\$E'T'$	$\$$	
$\$E'$	$\$$	
$\$$	$\$$	

b. Pembenahan Tingkat Frasa

Pembenahan tingkat frasa diimplementasikan dengan cara mengisi masukan kosong dalam tabel penguraian prakira dengan suatu *pointer* yang menunjuk pada *rountine* kesalahan. *Rountine* ini bisa menghapus, menyisipkan, mengubah simbol pada masukan serta memunculkan pesan kesalahan yang sesuai. *Rountine* ini juga dapat mem-*pop* sesuatu dari *stack*. Yang menjadi permasalahan disini adalah apakah kita harus memperbolehkan pengubahan suatu simbol *stack* atau mem-*push* simbol baru ke dalam *stack*, karena langkah-langkah yang dilakukan untuk pengurai bisa tidak sesuai sama sekali dengan penurunan suatu kata dalam suatu bahasa. Yang jelas, kita harus memastikan bahwa tidak ada *looping* tak hingga, untuk itu kita harus memeriksa bahwa setiap aksi pembenahan dapat mengakibatkan dipakainya suatu simbol masukan atau *stack*-nya dipendekkan jika masukan sudah diperoleh.

C. SOAL LATIHAN/TUGAS

1. Diketahui: $S \rightarrow aB|bA$
   $A \rightarrow a|aS|bAA$
   $B \rightarrow b|bS|aBB$
   Berikan solusi untuk sintaksnya !
2. Diketahui grammar $G = \{I \rightarrow H|IH|IA, H \rightarrow a|b|c|...|z, A \rightarrow 0|1|2|...|9\}$ dengan I simbol awal. Bagaimanakah analisa sintaks untuk kalimat x23b.
3. Terdapat statement: (A+B)*(C+D), akan menghasilkan bentuk sintaks?
4. Diketahui: $E \rightarrow E + T|T, T \rightarrow T * F|F, F \rightarrow (E)|I$, yang jelas mengandung recursion untuk simbol E dan T?
5. Diketahui:
   a. $E \rightarrow E + E$
   b. $E \rightarrow E * E$
   c. $E \rightarrow (E)$
   d. $E \rightarrow id$
   Salah satu derivasi *right-most* untuk $id_1 + id_2 * id_3$ adalah:

D. REFERENSI

Alfred V. Aho, Monica S., Ravi Sethi, Jeffrey D. (2007). *Compilers "Priciples, Techniques, & Tools"*, 2nd Edition. Boston, San Francisco, New York.

Jean-Paul Tremblay, Paul G. Sorenson. (1995). *"The Theory and Practice of Compiler Writing"*, United State of America.

Kenneth C. Louden. (1997). "*Compiler Construction: Principles and Practice"*.

Thomas W. Parsons. (1992). *"Introduction to Compiler Construction"*. New York: Computer Science Press.

Jean-Paul Tremblay & Paul G. Sorenson. (1985). "*The Theory and Practice of Compiler Writing"*. United State of America: International Edition.

Hari Soetanto. (2004). "*Teknik Kompilasi"*. Fakultas Teknologi Informasi Universitas Budi Luhur.

Sukamdi. (1995). *"Merekayasa Interpreter (Sebuah Penerapan Teknik Kompilasi)"*. PT. Elex Media Komputindo. (Buku Pegangan Pendamping). Jakarta.

Firrar Utdiartomo. (2001). *"Teknik Kompilasi*". J&J Learning. (Buku Pegangan Pendamping). Yogjakarta.

Sumantri Slamet, Heru S.,. (1995). *"Teknik Kompilasi"*. PT. Elex Media Komputindo. Jakarta.

GLOSARIUM

Integer adalah sebuah tipe data yang mempresentasikan bilangan bulat.

Ambigu adalah kemampuan mengekspresikan lebih dari satu penafsiran.

Backtracking adalah suatu algoritma umum yang dipakai untuk menemukan solusi dari permasalahan komputasi.

Array adalah tipe data terstruktur yang dapat menyimpan banyak data dari tipe data dan nama yang sama.