LAPORAN TUGAS II

IF2220 TEORI BAHASA DAN OTOMATA

IMPLEMENTASI CFG PADA PENGENALAN SINTAKS PROGRAM SEDERHANA

DISUSUN OLEH :

Jeremia Jason Lasiman 13514021

Bervianto Leo Pratama 13514047

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2015

A. DESKRIPSI PERSOALAN

Buatlah sebuah aplikasi untuk mengenali apakah sintaks sebuah program sederhana itu benar atau tidak. Aplikasi akan membuka file yang berisi sintaks program dan kemudian memeriksa apakah semua sintaks pada program tersebut valid atau tidak. Jika tidak valid maka aplikasi akan mengeluarkan informasi baris yang menyebabkan ketidakvalidan sintaks program.

B. JAWABAN PERSOLAN

Mengenali kebenaran sintaks sebuah program dapat dilakukan dengan CFG melalui implementasi parser yang dapat digunakan. Parser yang saat ini digunakan menggunakan sistem top-down parser, atau juga dikenal sebagai LL parser. Hal pertama yang perlu diperhatikan pada LL parser ini adalah pada pembacaan program, scanner perlu membaca sintaks program dan mengubahnya menjadi token-token yang kemdian menjadi acuan untuk pengecekan sintaks.

Setelah scanner mengubah tiap kata menjadi sebuah token, token tersebut diproses. LL parser ini menggunakan stack dalam pembuatannya. Jika token bukan merupakan terminal, maka top dari stack akan di pop, kemudian push serangkaian kode program yang diprediksi merupakan kelanjutan sintaks program. Kondisi berhasil dan gagal dapat dilihat melalui kondisi stack atau dummy jika digunakan.

CFG :

NONTERMINAL	TERMINAL	CFG :
S (START)	b (begin)	S -> bBe | be
B (BODY)	e (end)	B -> I | D | F | M | K | vEH
I (IF)	i (if)	I -> iPhS+
+ (jika ada else)	h (then)	+ -> lS
D (DO)	l (else)	D -> dBwP
F (FOR)	d (do)	F -> (fvEctc)dS
M (INPUT)	w (while)	M -> n(v)
K (OUTPUT)	t (to)	K -> o(v) | o(H)
E (EQUAL)	x (<, >, <>, <=, >=)	E -> =
P (PERHITUNGAN)	a (+, -, *)	P -> (_L_)
_ (const/var)	=	_ -> c | v
L (operator Logic)	n (input)	L -> x | E
	o (output)	
	(	
	)	
	v (variabel)	
	c (constant)	

C. SOURCE CODE

Penjelasan source code dan subprogram yang digunakan:

1. Prosedur InputMatrixStack
   Prosedur ini menginisialisasi matriks yang berupa isi dari setiap transisi nonterminal. Matriks ini berisi perkiraan perubahan tiap nonterminal dan juga perubahan nonterminal menjadi terminal.
2. Prosedur Pushing
   Prosedur ini memudahkan untuk melakukan push, dengan memanggil prosedur ini, stack akan di push dengan sintaks program yang benar.
3. Prosedur Scanner
   Prosedur ini membaca file kode-kode program dan mengubahnya menjadi suatu token-token yang akan dikenali oleh parser. Scanner akan membaca per karakter, untuk mengenali suatu kode tertentu akan ada pembacaan secara traversal hingga menemukan karakter unik.
4. Main
   Main berisi kode utama program parser, jika memenuhi beberapa syarat akan terjadi perubahan pada stack. Jika Top merupakan nonterminal maka akan diubah menjadi sintaks sesuai dengan matriks stack yang sudah disediakan. Tiap terminal pada stack yang sudah sesuai dengan input akan dikeluarkan (pop) dan mengetahui apakah sudah benar atau tidak melalui dummy dan kondisi stack.

D. CONTOH MASUKAN DAN KELUARAN

Masukan berupa teks file.

1. Input dari

```
begin
        input(z) {menerima masukan z berupa nilai integer}
        do
                if (z<10) then
                begin
                        a=z*2
                        output(a)
                end
                else
                begin
                        output(z*10)
                end
                input(z)
        while (z < 999)
end
```

Output :
Scan Success
Output Success

2. Input dari

```
begin
        input(z) {menerima masukan z berupa nilai integer}
        do
                if (z<10) then
                begin
                        a=z*2
                        output(a)
                end
                else
                begin
                        output(z*10)
                end
                input(z)
        while (z < 999)
```

Output :
Scan Success
Compile failed
Expected end of file at line – 14

3. Input dari

```
begin
{

end
```

Output :
Scan Success
Compile failed
Expected end of file at line - 1