|  |
| --- |
| Lexical Analyzer |
|  |
| Dosen Pengampu: SAID AL FARABY  Kelompok: 6  Anggota Kelompok:   1. Yogi Wisesa 1301154282 2. Ari Fahrezi 1301154268 3. M. Ridho Rahmadinanto 1301154226 4. M. Irhas Al Hafidz 1301154240 |

[Pick the date]

**2016**

# Kelompok:

|  |  |  |
| --- | --- | --- |
| **No** | **NIM** | **Nama** |
|  | 1301154282 | Yogi Wisesa Chandra |
|  | 1301154268 | Ari Fahrezi |
|  | 1301154226 | M. Ridho Rahmadinanto |
|  | 1301154240 | M. Irhas Al Hafidz |

# Spesifikasi Program

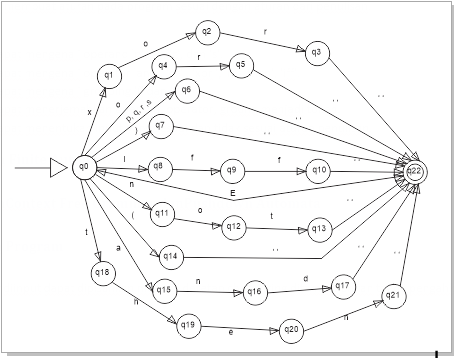
* Batasan Masalah

1. Mengenali operand, operator dan grouping.
2. Menggabungkan karakter- karakter yang tidak dipisahkan dengan spasi.
3. Menerjemahkan kata menjadi sebuah token.
4. Membuat aturan yang sesuai dengan ketentuan.

* Spesifikasi

1. Program dapat mengenali operand ‘p’, ’q’, ’r’, ’s’.
2. Program dapat mengenali operator ‘if’, ‘iff;, ‘xor’, ‘or’, ‘then’, ‘and’.
3. Program dapat mengenali grouping ‘(‘ dan ‘)’.
4. Program dapat menterjemahkan operand dan operator menjadi token.
5. Program dapat menuliskan inputan yang sesuai atau tidak sesuai.

# Rancangan Finite Automata



TABEL TRANSISI

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| State | p' | q' | r' | s' | n | o | t | a | d | r | " " | x | i | ( | ) | f | h | e |
| q0 | q6 | q6 | q6 | q6 | q11 | q4 | q18 | q15 | err | err | err | q1 | q8 | q14 | q7 | err | err | err |
| q1 | err | err | err | err | err | err | err | err | err | {] | {} | err | err | err | err | err | err | err |
| q2 | err | err | err | err | err | err | err | err | err | q3 | {} | err | err | err | err | err | err | err |
| q3 | err | err | err | err | err | err | err | err | err | {} | q22 | err | err | err | err | err | err | err |
| q4 | err | err | err | err | err | err | err | err | err | q5 | {} | err | err | err | err | err | err | err |
| q5 | err | err | err | err | err | err | err | err | err | err | q22 | err | err | err | err | err | err | err |
| q6 | err | err | err | err | err | err | err | err | err | err | q22 | err | err | err | err | err | err | err |
| q7 | err | err | err | err | err | err | err | err | err | err | q22 | err | err | err | err | err | err | err |
| q8 | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | q9 | err | err |
| q9 | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | q10 | err | err |
| q10 | err | err | err | err | err | err | err | err | err | err | q22 | err | err | err | err | err | err | err |
| q11 | err | err | err | err | err | q12 | err | err | err | err | err | err | err | err | err | err | err | err |
| q12 | err | err | err | err | err | err | q13 | err | err | err | err | err | err | err | err | err | err | err |
| q13 | err | err | err | err | err | err | err | err | err | err | q22 | err | err | err | err | err | err | err |
| q14 | err | err | err | err | err | err | err | err | err | err | q22 | err | err | err | err | err | err | err |
| q15 | err | err | err | err | q16 | err | err | err | err | err | err | err | err | err | err | err | err | err |
| q16 | err | err | err | err | err | err | err | err | q17 | err | err | err | err | err | err | err | err | err |
| q17 | err | err | err | err | err | err | err | err | err | err | q22 | err | err | err | err | err | err | err |
| q18 | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | q19 | err |
| q19 | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | q20 |
| q20 | err | err | err | err | q21 | err | err | err | err | err | err | err | err | err | err | err | err | err |
| q21 | err | err | err | err | err | err | err | err | err | err | q22 | err | err | err | err | err | err | err |
| q22 | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err | err |

PSEUDO CODE

Program Lexical;

Kamus

sText, token: string

i, length : integer

Algoritma

input(sText);

length <- sText.length

Space:

if i < length then

depend on sText(i)

Case "p" or "q" or "r" or "s": ' PQRS

i <- i + 1

depend on sText.Chars(i)

Case " "

i <- i + 1

token <- token + " 1" ' Token dikeluarkan saat sText.chars(i) <- " " agar mengetahui apakan suatu sintaks tidak ada lanjutannya, misal ifg agar menghasilkan error bukan 6error

GoTo Space

Case Else

token <- token + "error"

Case "n" ' NOT

i <- i + 1

depend on sText.Chars(i)

Case "o"

i <- i + 1

depend on sText.Chars(i)

Case "t"

i <- i + 1

depend on sText.Chars(i)

Case " "

token <- token + " 2"

i <- i + 1

GoTo Space

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case "a" ' AND

i <- i + 1

depend on sText.Chars(i)

Case "n"

i <- i + 1

depend on sText.Chars(i)

Case "d"

i <- i + 1

depend on sText.Chars(i)

Case " "

token <- token + " 3"

i <- i + 1

GoTo Space

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case "o" ' OR

i <- i + 1

depend on sText.Chars(i)

Case "r"

i <- i + 1

depend on sText.Chars(i)

Case " "

i <- i + 1

token <- token + " 4"

GoTo Space

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case "x" ' XOR

i <- i + 1

depend on sText.Chars(i)

Case "o"

i <- i + 1

depend on sText.Chars(i)

Case "r"

i <- i + 1

depend on sText.Chars(i)

Case " "

i <- i + 1

token <- token + " 5"

GoTo Space

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case "i" ' IF

i <- i + 1

depend on sText.Chars(i)

Case "f"

i <- i + 1

depend on sText.Chars(i)

Case " "

i <- i + 1

token <- token + " 6"

GoTo Space

Case "f"

i <- i + 1

depend on sText.Chars(i)

Case " "

i <- i + 1

token <- token + " 8"

GoTo Space

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case "t" ' THEN

i <- i + 1

depend on sText.Chars(i)

Case "h"

i <- i + 1

depend on sText.Chars(i)

Case "e"

i <- i + 1

depend on sText.Chars(i)

Case "n"

i <- i + 1

token <- token + " 7"

depend on sText.Chars(i)

Case " "

i <- i + 1

GoTo Space

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case Else

token <- token + "error"

Case "(" ' GROUPING

i <- i + 1

depend on sText.Chars(i)

Case " "

token <- token + " 9"

i <- i + 1

GoTo Space

Case Else

token <- token + "error"

Case ")" ' GROUPING JAGA- JAGA KALAU ADA GROUPING SETELAH GROUPING

i <- i + 1

depend on sText.Chars(i)

Case " "

token <- token + " 10"

i <- i + 1

GoTo Space

Case Else

token <- token + "error"

Case " "

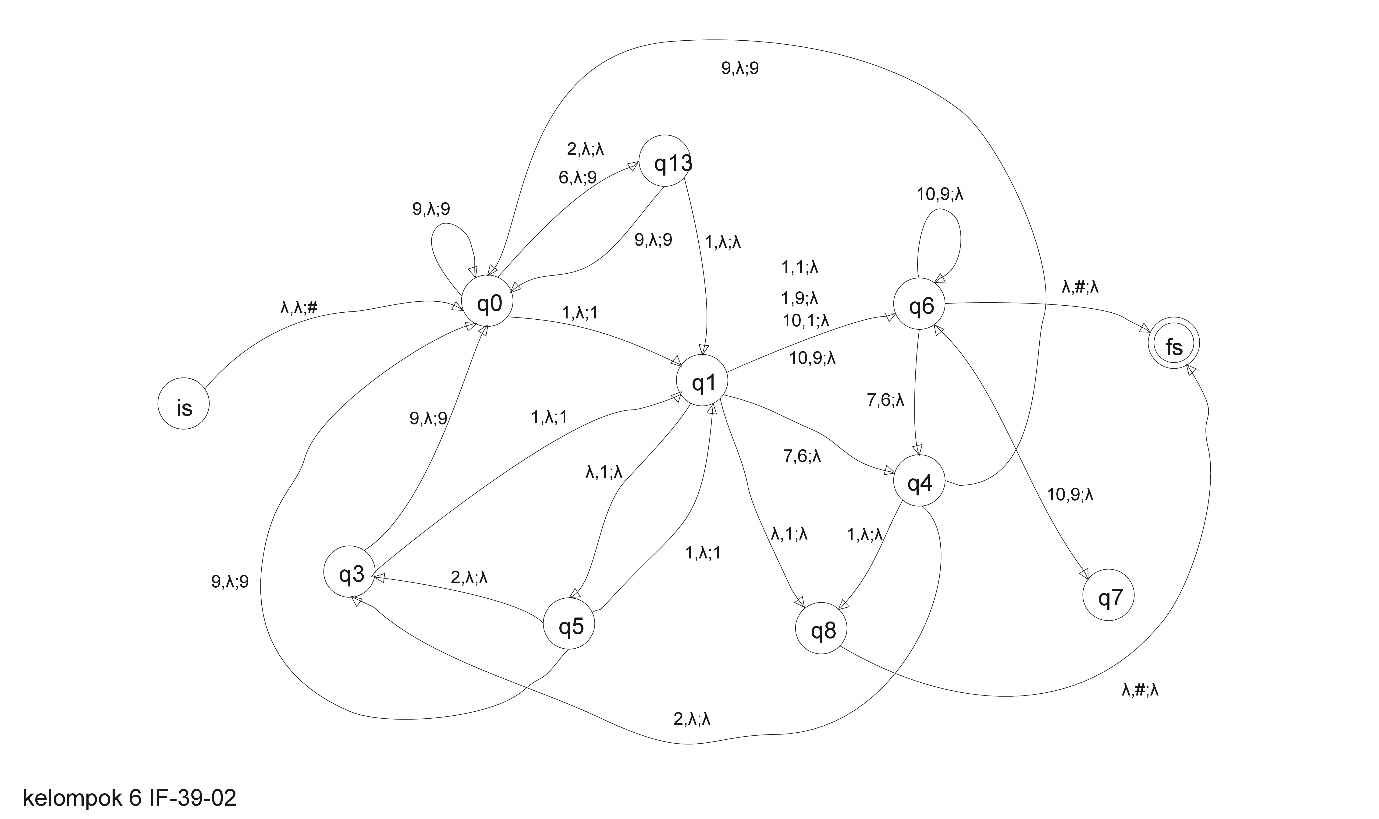
token <- token

Case Else

token <- token + "error"

output(token)

# Rancangan Context Free Grammar dan Pushdown Automata



# Cara Kerja Program

* Inputan.

Pada tahap 1 ini setiap token pada string tersebut dapat dikenali dengan inputan dari user yang di pisahkan dengan spasi (“ “). Program akan membaca setiap karakter dalam string dan dicocokan sesuai operator, operand maupun grouping yang telah ditentukan dan ketika karakter yang dicocokan merupakan spasi(" ") maka program akan memasukan token sesuai ketentuan ke dalam sebuah string yang nantinya akan dioutputkan ketika semua inputan telah terbaca.

* Proses output

Untuk inputan yang diterima benar program akan mengoutputkan token sesuai dengan ketentuan, jika terdapat kesalah dalam inputan maka program akan mengoutputkan “error” dan langsung berhenti memeriksa inputan, dengan ketentuan:

1. ‘p’ ‘q’ ‘r’ ‘s’ = 1
2. Not = 2
3. And = 3
4. Or = 4
5. Xor = 5
6. If = 6
7. Then = 7
8. Iff = 8
9. ( = 9
10. ) = 10
11. Lain = error

# Pengujian Program

Tuliskan hasil uji terhadap program yang dibangun. Berikan contoh beberapa input string arithmetic expression, kemudian tuliskan output yang dihasilkan program. Jelaskan hasil output tersebut.

|  |  |  |
| --- | --- | --- |
| **Input** | **Output Program** | **Keterangan** |
| p and q or r | 1 3 1 4 1 | Sesuai |
| p xor ( q and not ( p and q ) ) | 1 5 9 1 3 2 9 1 3 1 10 10 | Sesuai |
| ( p and q ifg ( r or s ) | 9 1 3 1 error | Sesuai |

# Screenshot Program

Tunjukkan beberapa screenshot program yang telah dibangun.

