Assignment 2

Laraib jahangir

l16-4052

2020

List of lexemes are as follows:

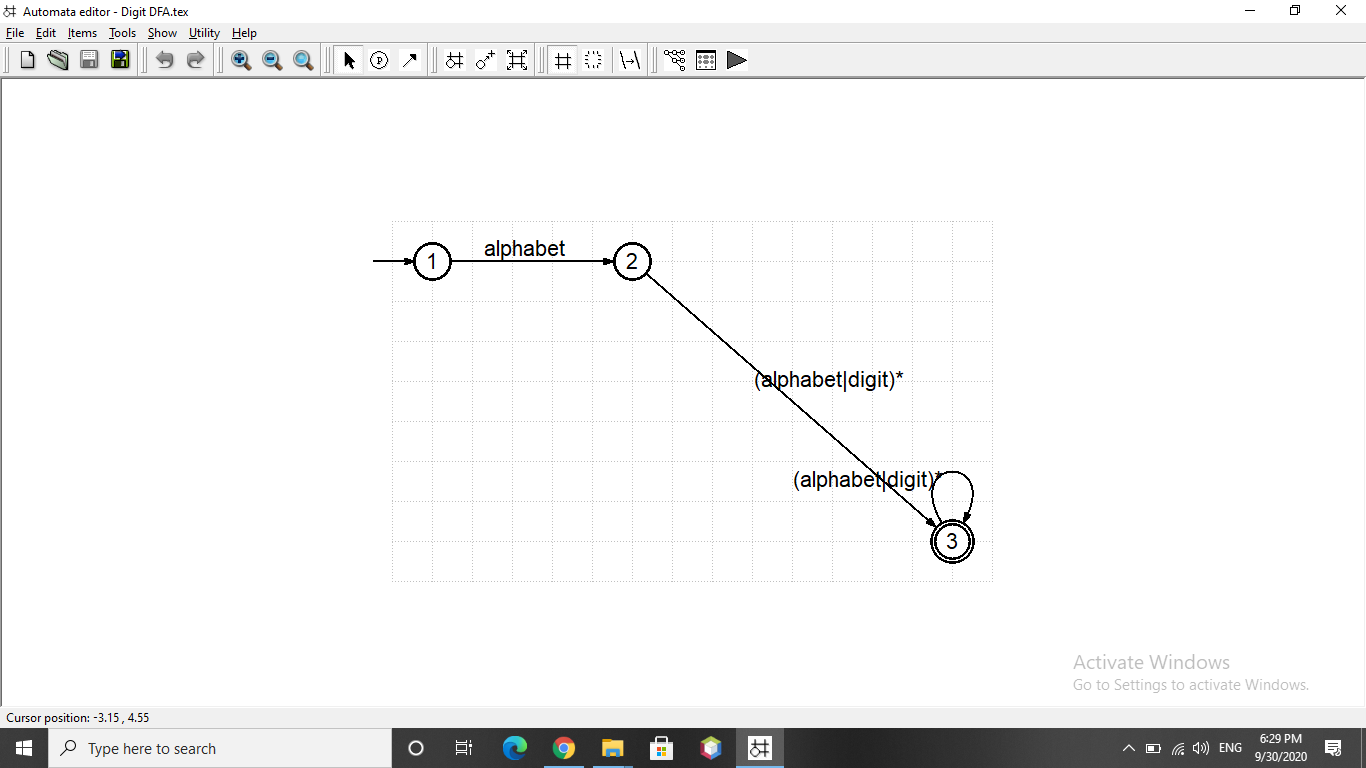
1. datatypes: int , char
2. Keywords:if, else, while,ret,write,read
3. Arithmetic operators: +,-,\*,/
4. Relational operator: <,<=,>,>=,=,/=
5. Comments-> \*./|(string|spaces)\*.\*./
6. Numeric constant:only integers
7. Decleration operator: :
8. Literal constant:a letter enclosed in a single quote
9. parenthisis, braces, square brackets
10. Assignment operator: :=
11. Input operator: >>
12. Semicolon,comma,colon
13. Idetifier -> (alphabet).(alphabets| Digits)\*

The description of language tokens for a given program is as follows:

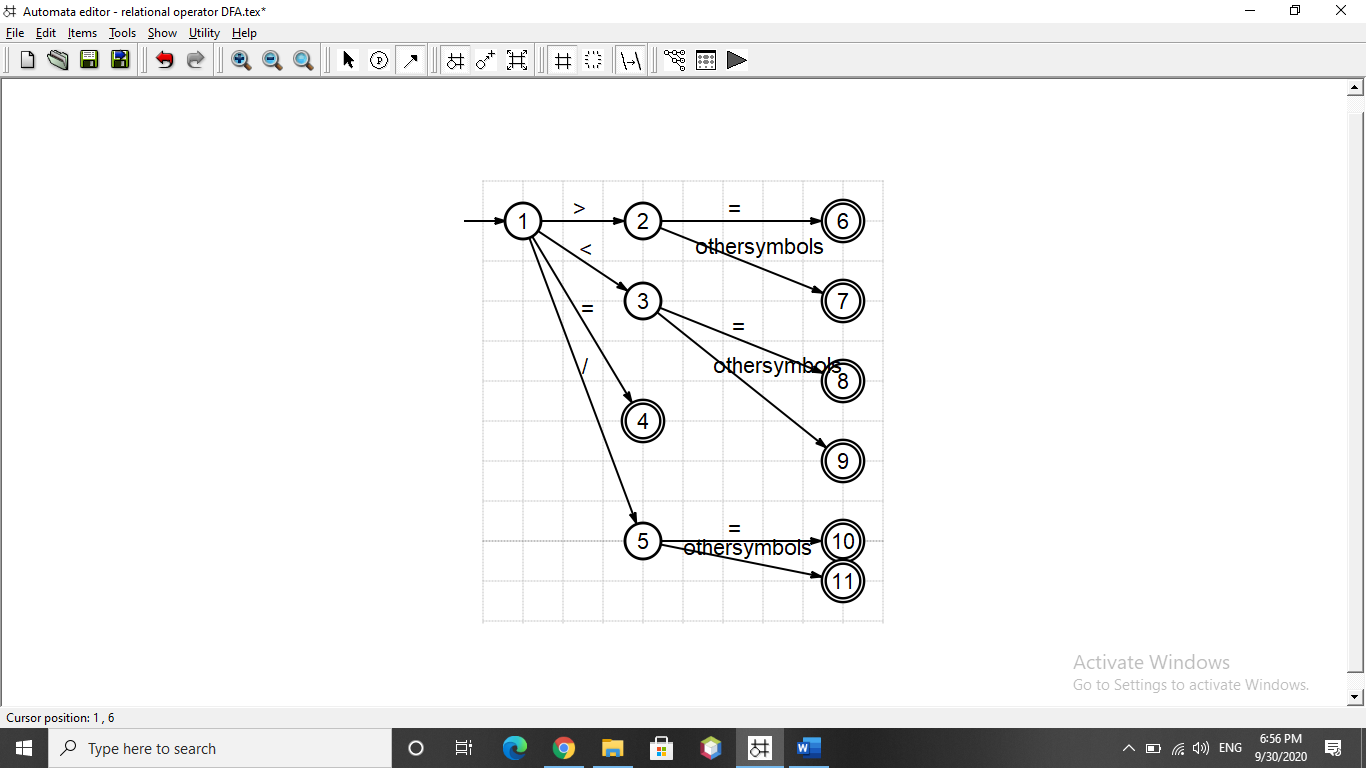
|  |  |  |
| --- | --- | --- |
| **Regular Expression** | **Token** | **Attribute value** |
| Int | Datatype | Int |
| : | Declaration operator | - |
| num | Identifier | - |
| ( | Parenthesis | - |
| int | Datatype | int |
| : | Declaration operator | - |
| Num | Identifier | Look into symbol table |
| , | Comma | - |
| Int | Datatype | int |
| : | Declaration operator | - |
| length | Identifier | Look into symbol table |
| { | Braces | - |
| : | Declaration operator | - |
| i | Identifier | Look into symbol table |
| J | identifier | Look into symbol table |
| First | Identifier | Look into symbol table |
| temp | identifier | Look into symbol table |
| ; | Semi colon | - |
| Char | Datatype | Char |
| : | Declaration operator | - |
| a | Identifier | Look into symbol table |
| ; | Semi colon | - |
| a | Identifier | Look into symbol table |

|  |  |  |
| --- | --- | --- |
| := | Assignment operator | - |
| ‘x’ | Literal constant | - |
| ; | Semi colon | - |
| write | Keyword | Write |
| ( | Parenthesis | - |
| ) | Parenthesis | - |
| ; | Semi colon | - |
| Read | Keyword | Read |
| >> | Input operator | - |
| ( | Parenthesis | - |
| i | Identifier | Look into symbol table |
| ) | Parenthesis | - |
| ; | Semi colon | - |
| i | Identifier | Look into symbol table |
| := | Assignment operator | - |
| length | Identifier | - |
| .  .  . | | |
| } | Braces | - |
| /\* | Comments | - |
| \*/ | comments | - |
| .  .  . | | |

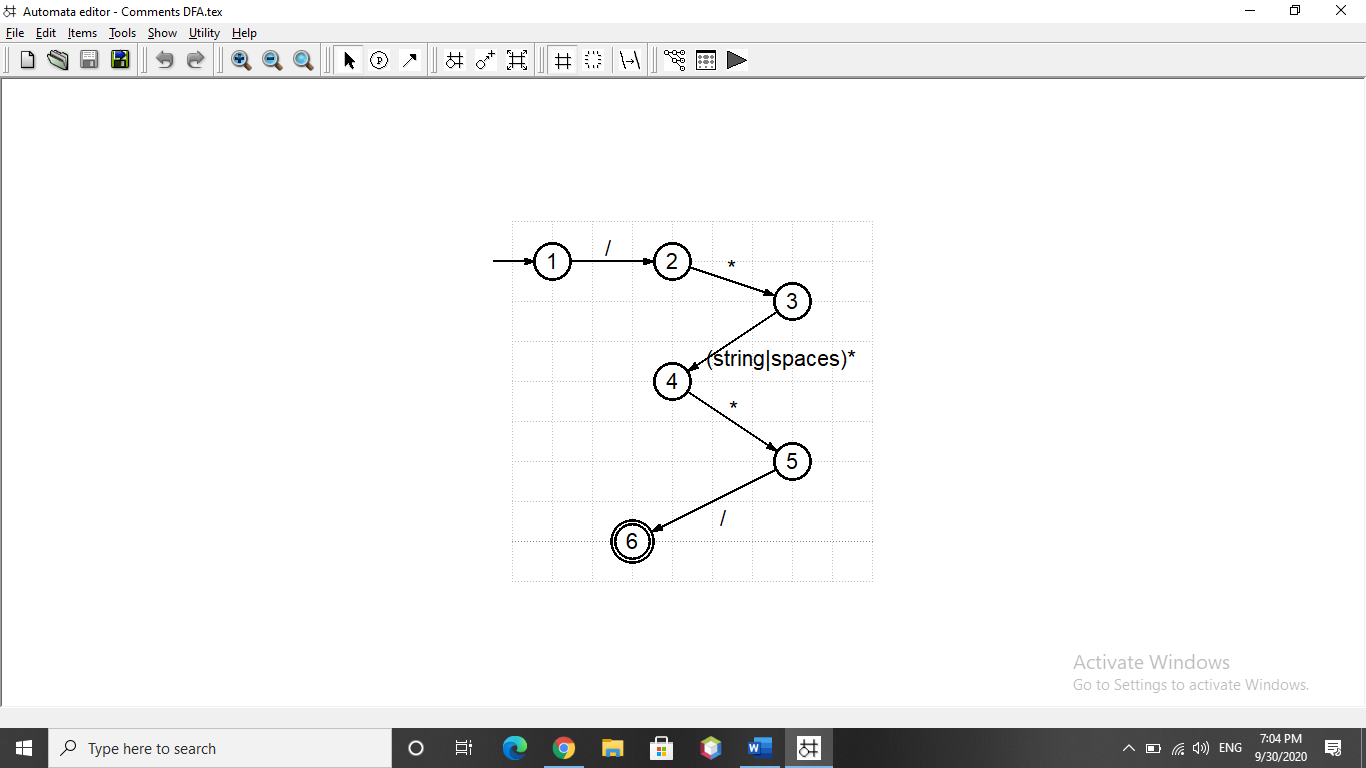
DFA for identifier, comments, and relational operators are provided.



**DFA for identifiers**

****

**DFA for relational operators**

****

**DFA for comments**

**Parser for language**

We do following steps to design a parser:

* Write CFG.
* Rewrite CFG
  + Set associativity
  + Set precedence
  + Remove ambiguity
  + Remove left recursion

P -> V | F ( p for program , v for variable , f for function)

V -> DT . id (DT for datatypes)

DT -> int | char

F -> DT . id . (PL) . {SL} (pl for parameter list , sl for statement list)

PL -> PA | PL | ^ (pa for parameter , pl for parameter list)

PA -> DT . : . id

SL -> S | SL | ^ (sl for statement list)

S -> AS | RS | BS | IS (As for assignment statement, rs for relational statement , Bs for branch statement , IS for iterative statement )

RS -> id | E . > . id | E | id | E .>= . id | E | id | E . < . id | E | id | E . < =. id | E | id | E . < /= . id | E

AS -> id. := . E

BS -> IF(E) {SL} | else {SL}

IS -> WHILE ( E ) {SL}

id -> Alphabet | id. Alphabet |id. Digit

E -> E + T | E-T | E \* T | E / T |(E) | Digit | Alphabet

Digit -> 0| 1 | 2 | … | 9

Alphabet -> a | b | c | …… | z | A| B | C |…. | Z

**Setting precedence:**

Brackets have highest precedence (highest)

Asterik and division has same precedence (higher)

Addition and subtraction has least precedence (least)

E -> E+ T | E-T | F

F -> F \* T | F / T | B

B-> (E ) | digit | alphabet

**Removing left recursion:**

E -> F R1 | digit R1 | alphabet R1

R1 -> + T R1 | - TR1

R1 -> ^

F -> BR2 | \* T R2 | / TR2 | (E)

R2 -> ^