Source code
(character stream)

if (b == 0) a = b;

Token stream

Identifiers: x y11 elsen _i00
Integers: 2 1000 -500 5L
Floating point: 2.0 .02 1. 1e5 0.e-10
Strings: "x" "She said, "Hey!"
Comments: /** don’t change this **/
Keywords: if else while break
Symbols: + * { } ++ < << [ ] >=

Lexical Analysis
Regular Expressions

A language is a set of words: \{ moo, cow \}, \{ a,b,c,d,... \}

Regular expressions describe languages

\[ abab \quad a|b \quad (a|b)^* \quad [1-9][0-9]^* \quad [a-z][a-z0-9]^* \]

Definition

\( a \) ordinary character stands for itself

\( \varepsilon \) the empty string

\( R|S \) either \( R \) or \( S \) (alternation), where \( R,S \) are REs

\( RS \) \( R \) followed by \( S \) (concatenation)

\( R^* \) \( R \) repeated 0 or more times

\( L(R) = \) the language defined by regular expression \( R \)

\( L(a (moo | cow)) = \{ amoo, acow \} \)

\( L([1-9][0-9]^*) = \{ 1,2,3,4,5,6,7,8,9,10,11,12,13,.... \} \)
Acceptors:
(a.k.a. recognizers)

Abstract machines that determine if an input string belongs to a language, answering Yes/No.

Finite Automata:
acceptors for languages described by regular expressions
Finite Automata

Regular Expression: \((-|\varepsilon)[0-9][0-9]^*\)

Non-deterministic Finite Automata:

Deterministic Finite Automata:
Building an acceptor for a regular expression:

Regular Expression $R$ -> $\text{RE }\Rightarrow\text{ NFA Conversion}$

$\text{NFA }\Rightarrow\text{ DFA Conversion}$

$\text{DFA Simulation}$

Input String $w$ ->

Yes, if $w \in L(R)$
No, if $w \notin L(R)$