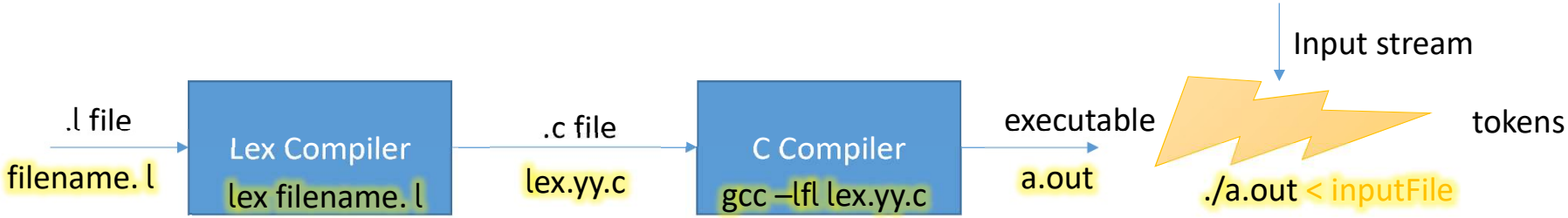
Lex

Rule based Programming Language

Usage


.l file
filename. l
Lex Compiler
lex filename. l
.c file
lex.yy.c
C Compiler
gcc –lfl lex.yy.c
executable
a.out
Input stream
./a.out < inputFile
tokens

Structure of a Lex Program

Declaration

%%

Translation rules

%%

Auxiliary Procedures

Translation Rules

Pattern(Regular Expression) Action

Eg.

1 {printf(“one”);}

Operators and Characters in the Rules

Meta Character	Matches
.	Any character except new line
\n	New line
[]	Character class [any one character within] – eg. [xy], [x-z]
[^x]	Any character but x
x*	0 or more occurrences of the preceding expression x
x+	1 or more occurrences of the preceding expression x
x?	0 or 1 occurrence of the preceding expression x
^x	Line beginning with x
x$	Line ending with x
a|b	Expression a or expression b
(ab)+	1 or more copies of “ab” together
“a+b”	Literal “a+b” [interpreted as is]
\x	x, if x is a lex operator – eg. \{, \[, *
x{m,n}	m to n occurrences of the preceding expression x

Pattern Matching Examples

Expression	Matches
`abc`	`abc`
`abc*`	`ab abc abcc abccc ...`
`abc+`	`abc abcc abccc ...`
`a(bc)+`	`abc abcbc abcbcbc ...`
`a(bc)?`	`a abc`
`[abc]`	one of: `a`, `b`, `c`
`[a-z]`	any letter, a-z
`[a\-z]`	one of: `a`, `-`, `z`
`[-az]`	one of: `-`, `a`, `z`
`[A-Za-z0-9]+`	one or more alphanumeric characters
`[ \t\n]+`	whitespace
`[^ab]`	anything except: `a`, `b`
`[a^b]`	one of: `a`, `^`, `b`
`[a|b]`	one of: `a`, `|`, `b`
`a|b`	one of: `a`, `b`

Declaration

Pattern

name	pattern
Eg.	
space	[\t]
ws	{space}+
letter	
digit	
cVariable	

C

- Variables
- Functions
- Header file inclusion

Eg.

```
%{
        int count = 0;
%}
```

Auxiliary Procedures

- C functions

Predefined Functions and Variables

Name	Function
`int yylex(void)`	call to invoke lexer, returns token
`char *yytext`	pointer to matched string
`yyleng`	length of matched string
`yylval`	value associated with token
`int yywrap(void)`	wrapup, return 1 if done, 0 if not done
`FILE *yyout`	output file
`FILE *yyin`	input file
`INITIAL`	initial start condition
`BEGIN`	condition switch start condition
`ECHO`	write matched string

Default Program with %%

```
%%
    /* match everything except newline */
.   ECHO;
    /* match newline */
\n  ECHO;

%%

int yywrap(void) {
    return 1;
}

int main(void) {
    yylex();
    return 0;
}
```

Sample Program – count number of lines

```
%{
        int numOfLines = 0;
%}
%%
\n      numOfLines++;
.       ;
%%
void main(){
        yylex();
        printf("Number of lines = %d", numOfLines);
}
```