Compiler Design Laboratory (CS 753)

Samit Biswas

samit@cs.iiests.ac.in

Department of Computer Science and Technology,
Indian Institute of Engineering Science and Technology, Shibpur

August 20, 2018

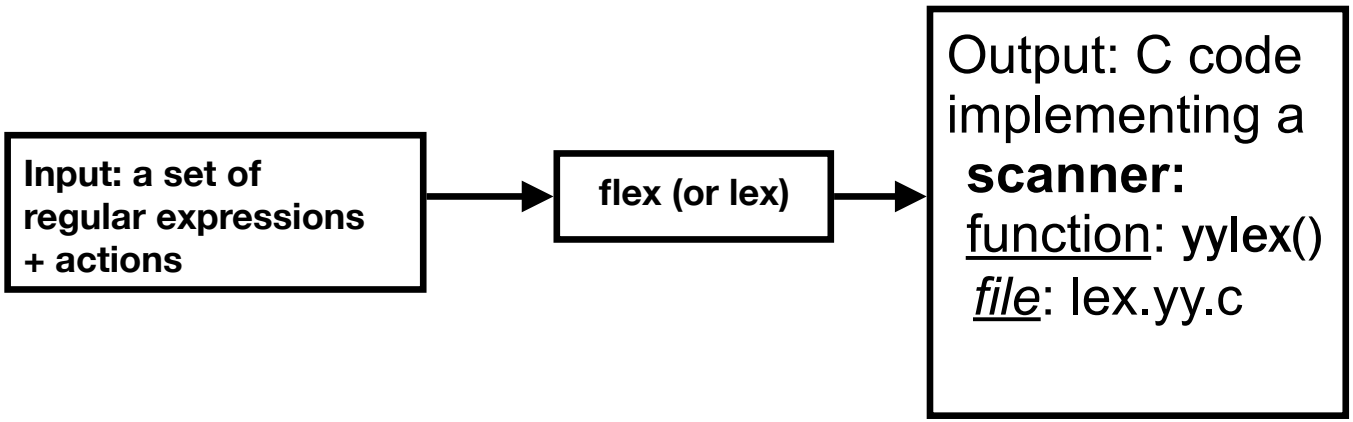
lex/flex

- We refer to the tool as Lex compiler , and to its input specification as the Lex language.

flex (and lex): Overview

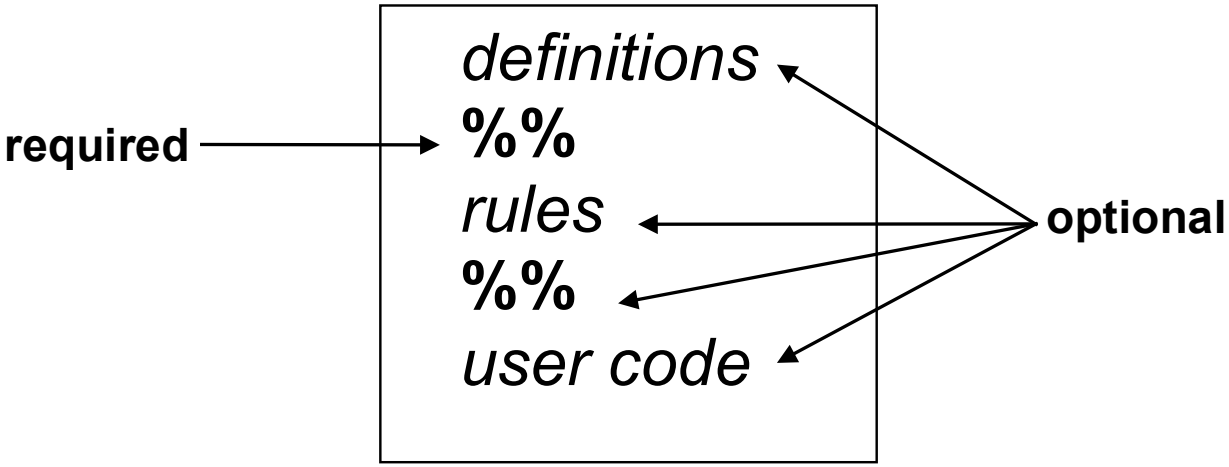
Scanner generator

- Helps write programs whose control flow is directed by instances of regular expressions in the input stream.



flex input format

A Lex program (the lex.l file) consists of three parts:



- Shortest possible legal flex input:

```
%%
```

Definitions

- name definitions, each of the form

  name *definition*

  e.g.:

- name definitions, each of the form

DIGIT	*[0-9]*
CommentStart	"/*"
ID	[a-zA-Z][a-zA-Z0-9]*

- stuff to be copied verbatim into the flex output enclosed in %{ ..... }%, or (e.g., declarations, #includes):

Lex Rules

- The rules portion of the input contains a sequence of rules.
- Each rule has the following form
  Regular_Expression *Action*
  where:
  - *Regular_Expression* describes a pattern to be matched on the input.
  - *action* must begin on the same line.

Example of Lex Rules

$$< Reg.Exp > < action >$$

Pattern	Action
int	printf(“Keyword: INTEGER ”);
[0-9]+	printf(“Number ”);

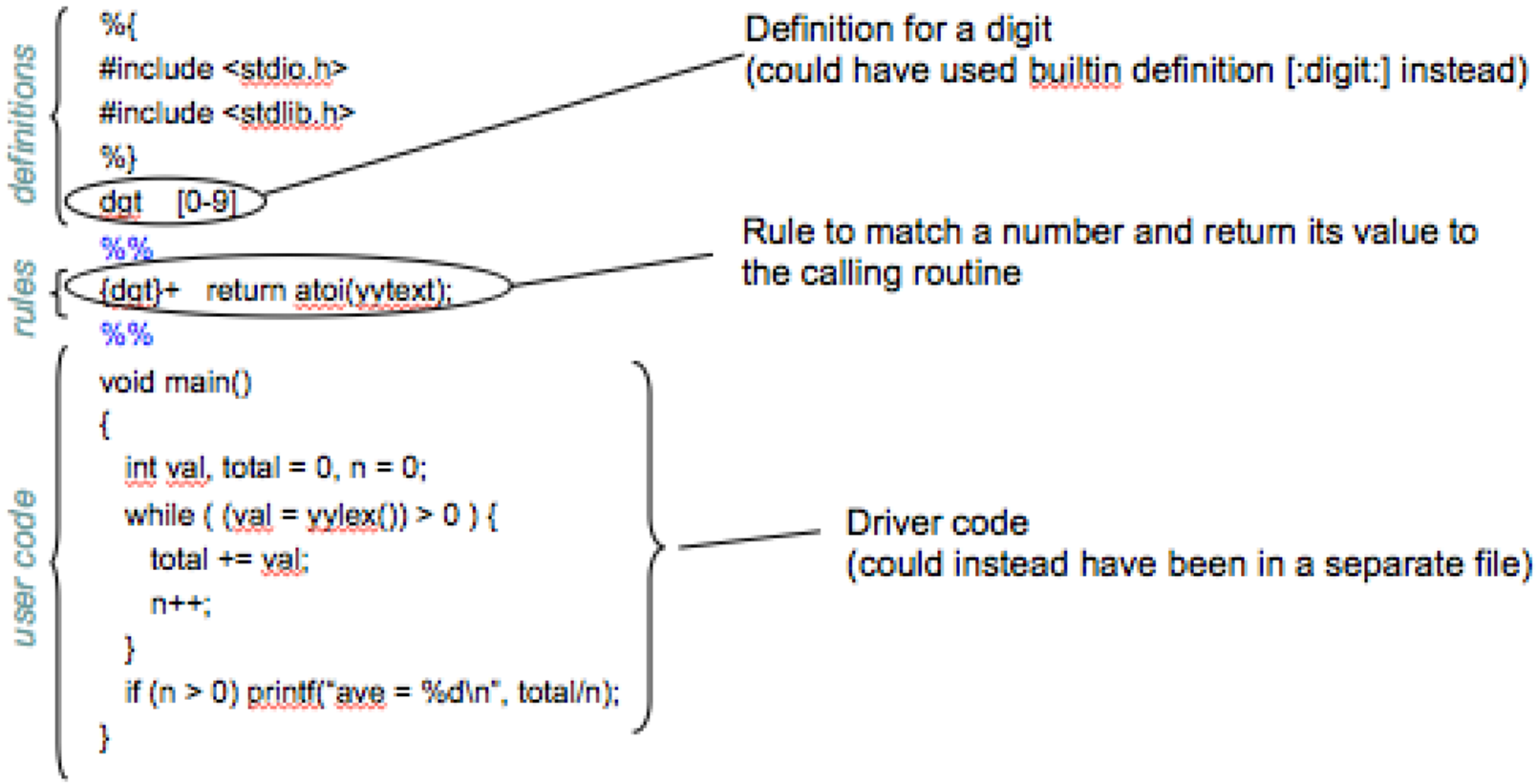
Patterns

- Essentially, extended regular expressions.
  - Syntax: similar to *grep* (see *man* page).
  - $<<$ *EOF* $>>$ to match "end of file".
- Character classes:

  [:alpha:], [:digit:], [:alnum:], [:space:], etc. (see man page).
- $\{name\}$ where name was defined earlier.

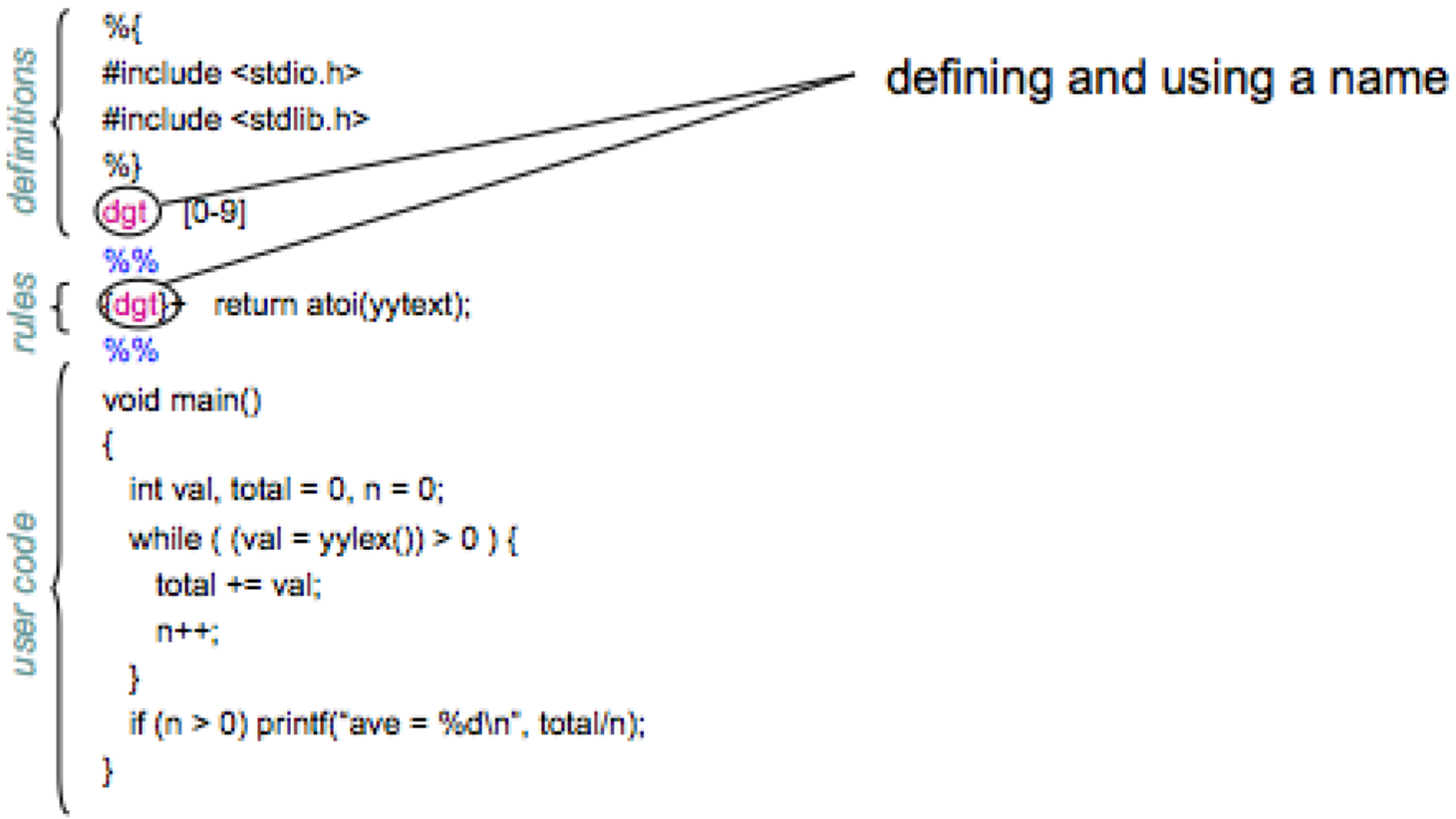
Example

A flex program to read a file of (positive) integers and compute the average:



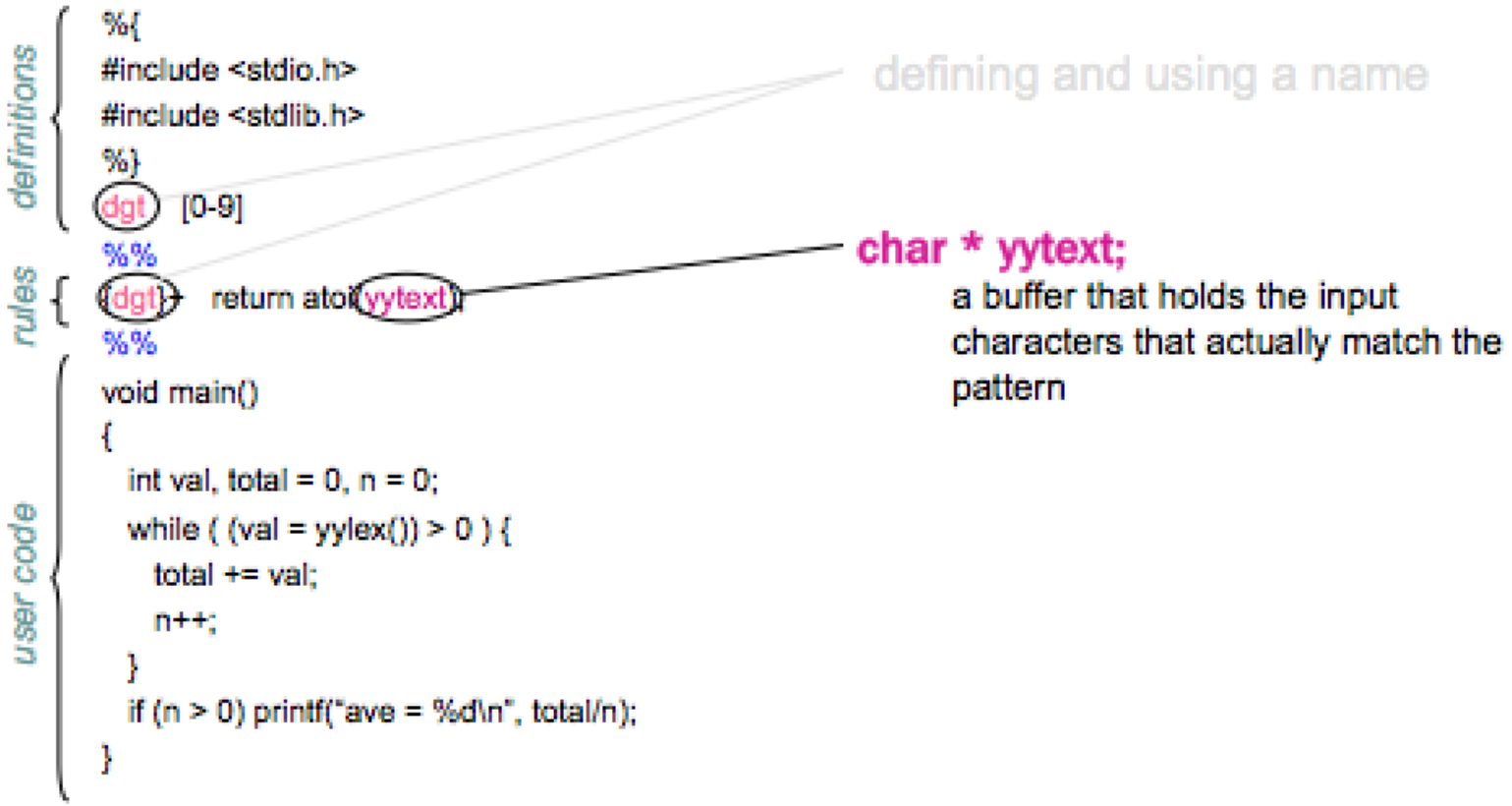
Example

A flex program to read a file of (positive) integers and compute the average:



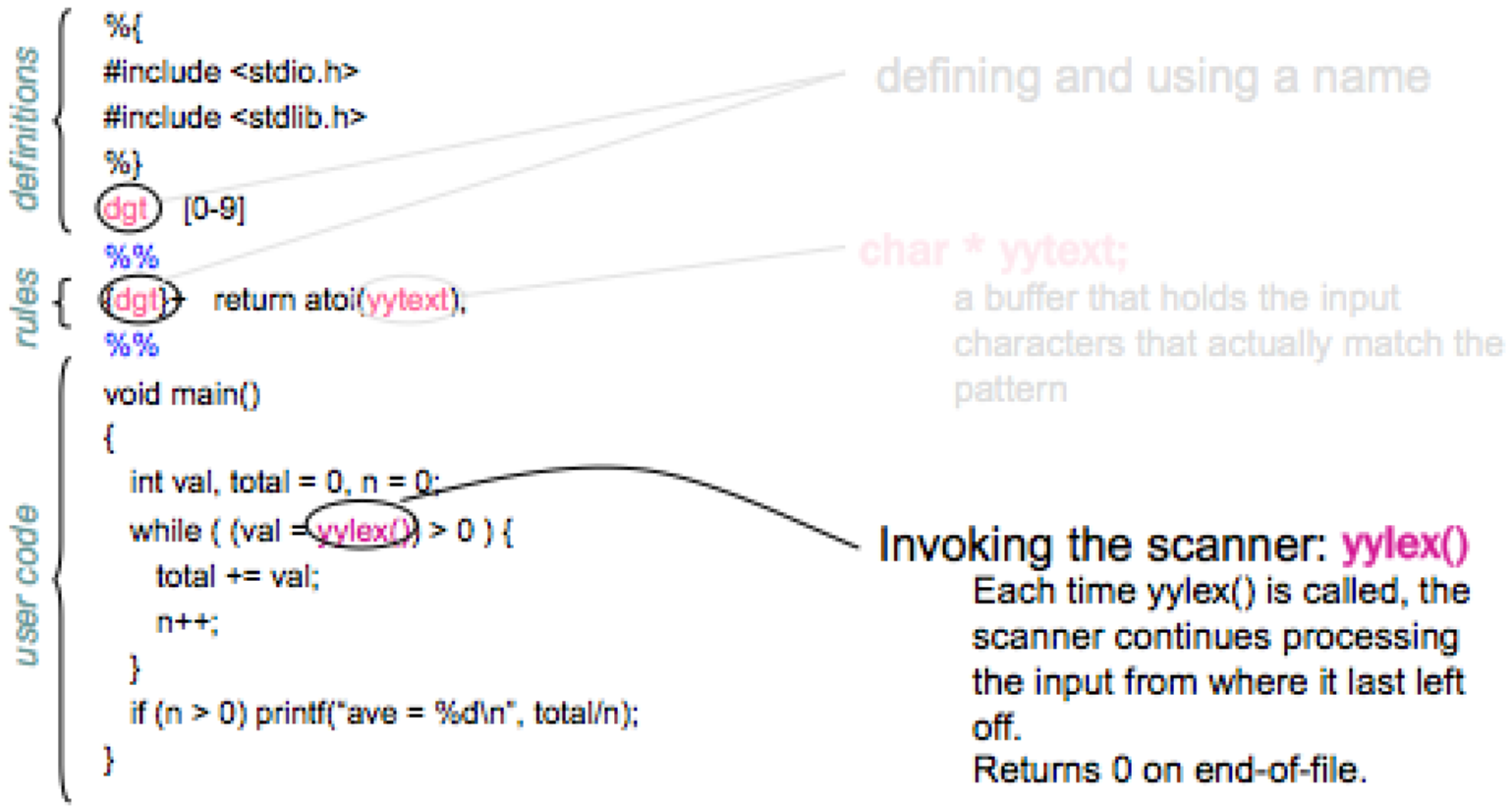
Example

A flex program to read a file of (positive) integers and compute the average:



Example

A flex program to read a file of (positive) integers and compute the average:



Lex library routines

- *yylex()*
  The default main() contains a call of yylex()
- *yymore()*
  return the next token.
- *yyless(n)*
  retain the first n characters in yytext.
- *yywrap()*
  - is called whenever Lex reaches an end-of-file.
  - The default yywarp() always returns 1.

Lex Predefined Variables

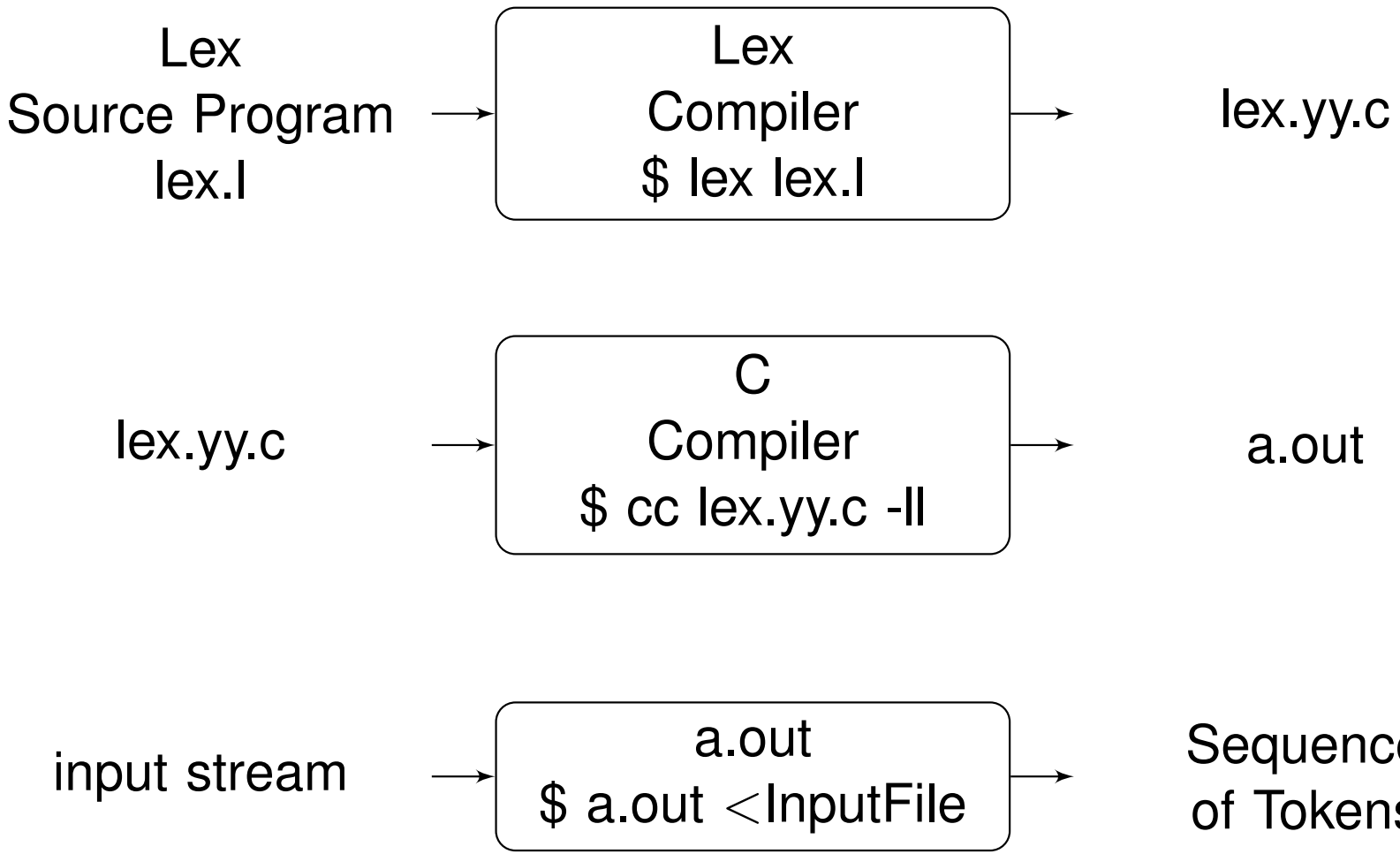
- *yytext* – a string containing the lexeme
- *yyleng* – the length of the lexeme
- *yyin* – the input stream pointer
  - the default input of default main() is **stdin**
- *yyout* – the output stream pointer
  -the default output of default main() is **stdout**

In lex program, a main() function is generally included as:

```
main(){
yyin = fopen(filename, "r");
while(yylex());
}
```

- Here **filename** corresponds to input file and the *yylex*() routine is called which returns the tokens. *yyin* is *FILE* pointer declared by Lex part.

Lexical Analyser Generators — Lex



Assignment

Implement a lexical analyzer using the tool: lex/flex for the following types of tokens:

- Arithmetic, Relational, Logical, Bitwise and Assignment Operators of C.
- Reserved words: int, float, char, for, while, if and else
- Identifier.
- Integer Constants.
- Parentheses, Curly braces

Take a complete C program as input and generate the above-mentioned tokens.

tokendef_LEX.h

```
/* Single caharacter lexemes */
#define LPAREN_TOK '('
#define GT_TOK '>'
#define RPAREN_TOK ')'
#define EQ_TOK '='
#define MINUS_TOK '-'
#define SEMICOLON_TOK ';'
/*
.
.
.*/
/* Reserved words */
#define WHILE_TOK 256
/*
.
.
.*/
/* Identifier, constants..*/
#define ID_TOK 350
#define INTCONST 351
/*
.
.
.*/
```