# Compiler Design

## An Introduction

Samit Biswas[1]

[1]Department of Computer Science and Technology,
Indian Institute of Engineering Science and Technology, Shibpur
Email: samit@cs.iiests.ac.in

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# Table of Contents



IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# Introduction To Lex/Flex

**What is Lex?**

- A tool for building lexical analyzers (lexers)

# Introduction To Lex/Flex

**What is Lex?**

- A tool for building lexical analyzers (lexers)
- **lexer (scanner)** is used to perform lexical analysis, or the breaking up of an input stream into meaningful units, or tokens.
  - E.g., consider breaking a text file up into individual words.

# Introduction To Lex/Flex

**What is Lex?**

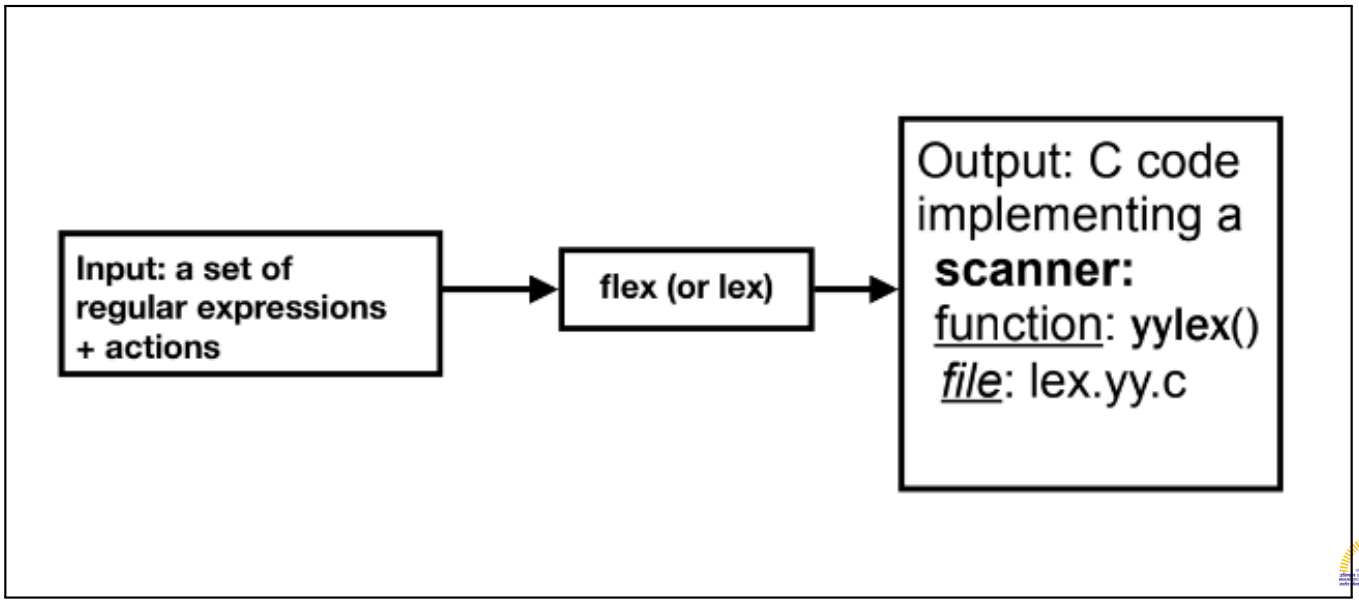
- A tool for building lexical analyzers (lexers)
- **lexer (scanner)** is used to perform lexical analysis, or the breaking up of an input stream into meaningful units, or tokens.
  - E.g., consider breaking a text file up into individual words.
- We refer to the tool as **Lex compiler**, and its input specification as the **Lex language**.

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# Introduction To Lex/Flex

**What is Lex?**

- A tool for building lexical analyzers (lexers)
- **lexer (scanner)** is used to perform lexical analysis, or the breaking up of an input stream into meaningful units, or tokens.
  - E.g., consider breaking a text file up into individual words.
- We refer to the tool as **Lex compiler**, and its input specification as the **Lex language**.
- **flex (and lex)**: Overview




IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

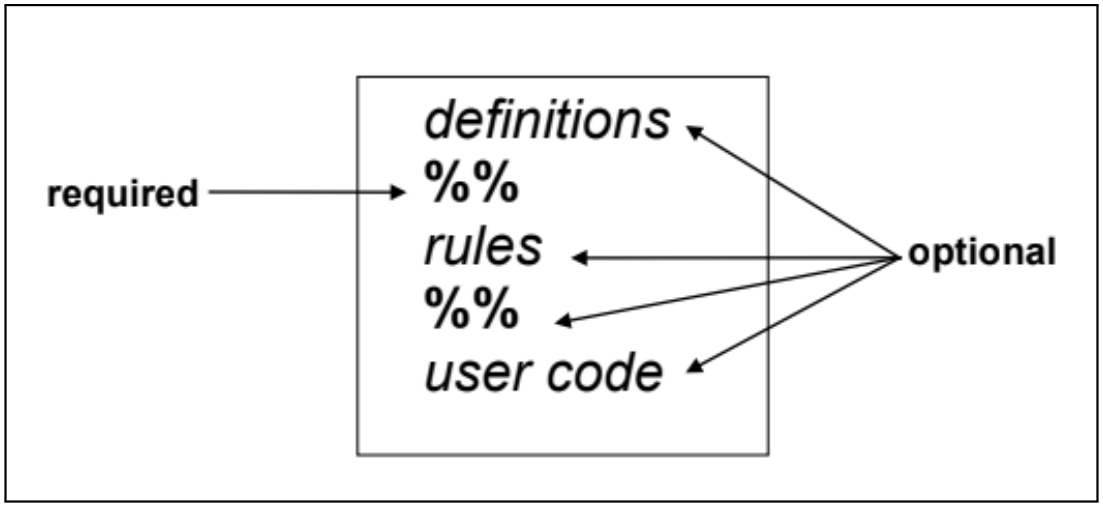
# Table of Contents



IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# Skeleton of a lex specification

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




- Shortest possible legal flex input:

```
%%
```

# Skeleton of a lex specification

| **Section** | **Purpose** |
|---|---|
| Definition Section | This part will be embedded into *.c |

# Skeleton of a lex specification

| **Section** | **Purpose** |
|---|---|
| Definition Section | This part will be embedded into *.c |
| Rules Section | define how to scan and<br>what action to take for each token |

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# Skeleton of a lex specification

| **Section** | **Purpose** |
|---|---|
| Definition Section | This part will be embedded into *.c |
| Rules Section | define how to scan and<br>what action to take for each token |
| User Code | any user code.<br>For example, a main function to<br>call the scanning function yylex(). |

# flex input format

**Definitions:**

- name definitions, each of the form

| **name** | **definition** |
|---|---|
| DIGIT | $[0-9]$ |

# flex input format

**Definitions:**

- name definitions, each of the form

| **name** | **definition** |
|---|---|
| DIGIT | $[0-9]$ |
| Comment Start | "/*" |

# flex input format

**Definitions:**

- name definitions, each of the form

| **name** | **definition** |
|---|---|
| DIGIT | $[0-9]$ |
| Comment Start | "/*" |
| ID | [a-zA-Z][a-zA-Z0-9]* |

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# flex input format

**Definitions:**

- name definitions, each of the form

| **name** | **definition** |
|---|---|
| DIGIT | $[0-9]$ |
| Comment Start | "/*" |
| ID | [a-zA-Z][a-zA-Z0-9]* |

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

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# 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.

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# Lex Rules

**Example of Lex Rules**

- $< Reg.Exp > < action >$

| **Pattern** | **Action** |
|---|---|
| int | printf("Key word: Integer"); |
| $[0-9]+$ | printf("Number"); |

# Patterns

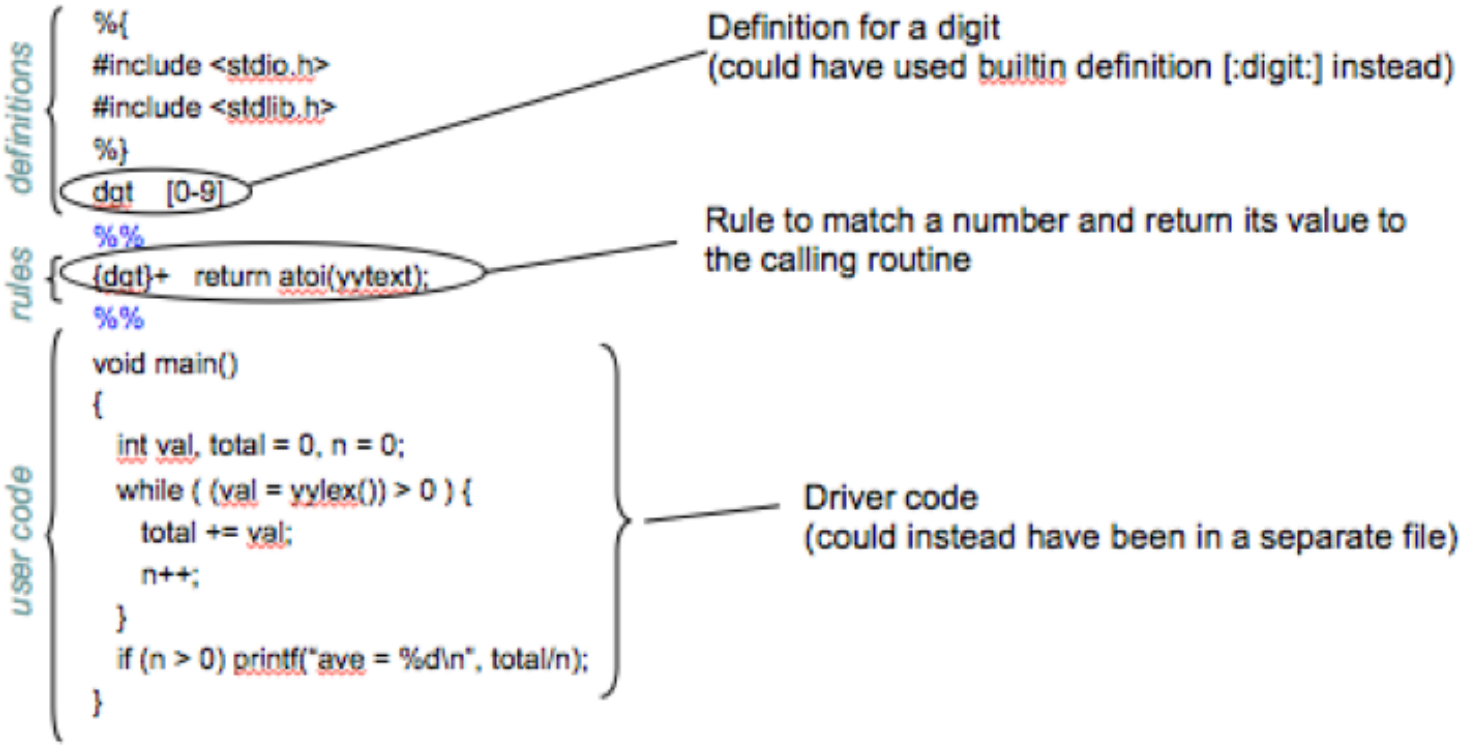
**Patterns**

- Essentially, extended regular expressions.
  - Syntax: similar to grep (see man page).
  - $<<$ *EOF* $>>$ to match the end of file.
- Character classes:
  - [: *alpha* :], [:digit:], [:alnum:], [:space:], etc. (see man page).
- {name} where name was defined earlier.

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

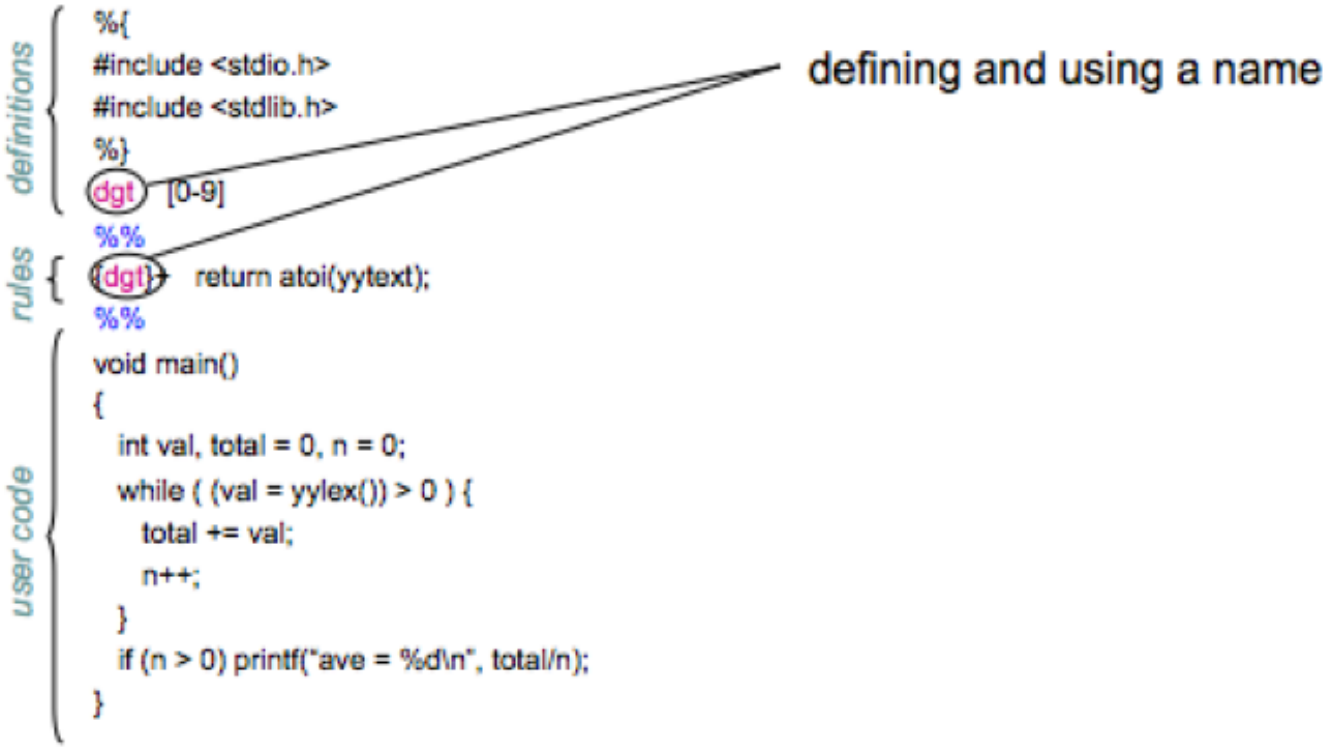
# Example

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



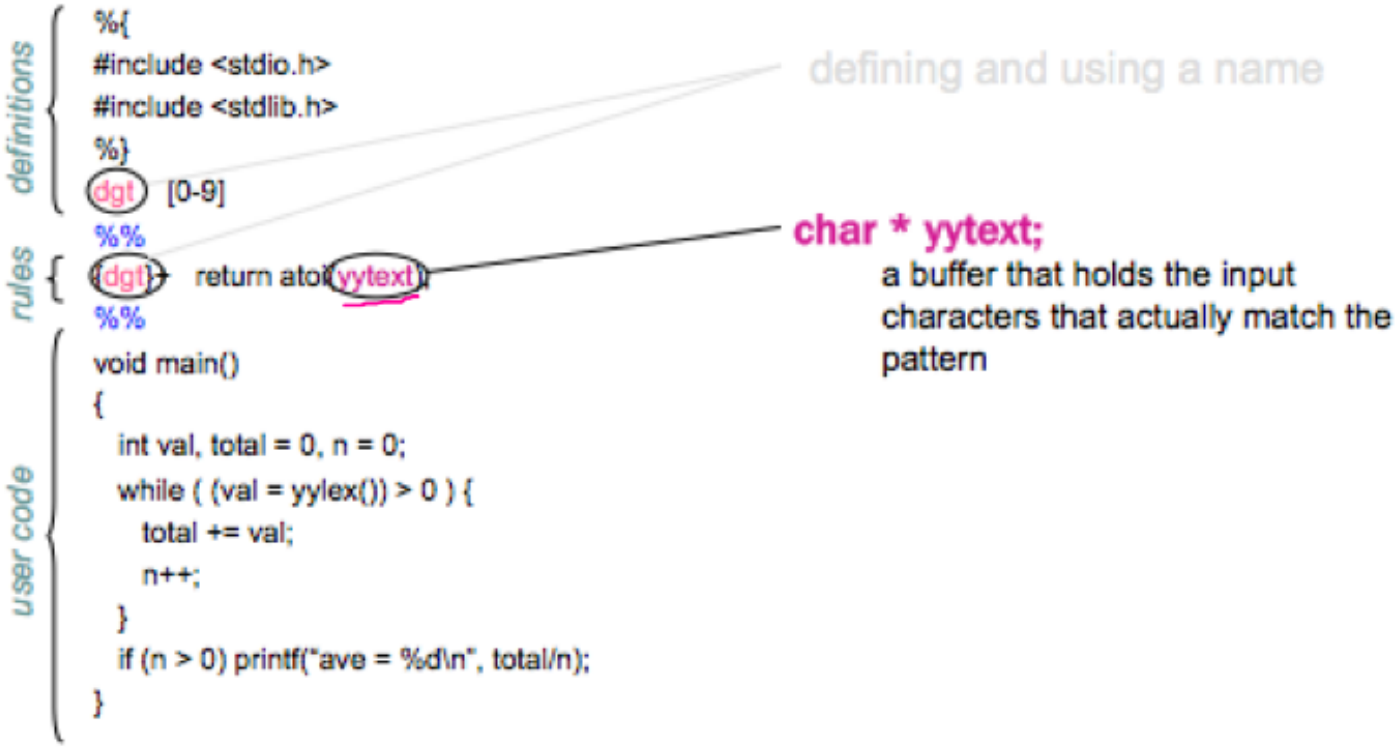
# Example

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



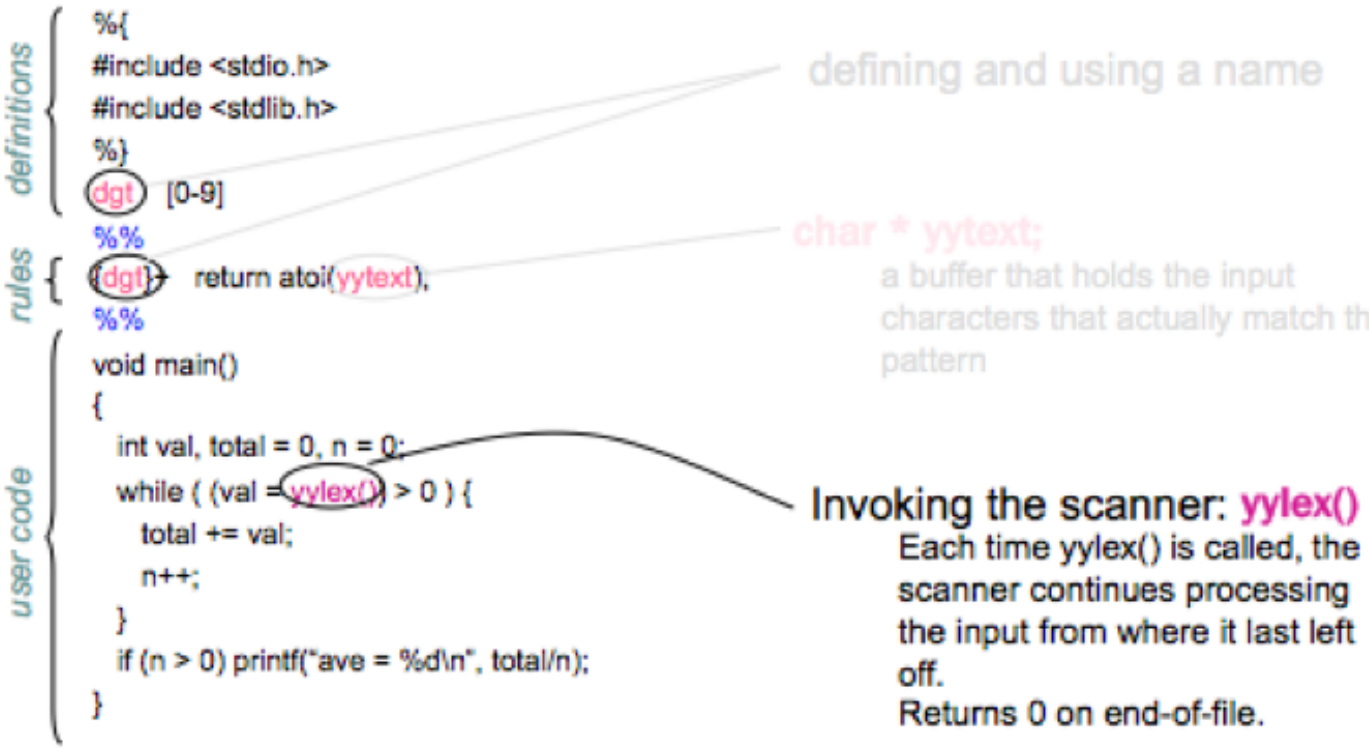
# Example

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:



# Table of Contents



IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# Lex library routines

## 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

## 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**

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# lex program, a main() function

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

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

# lex program, a main() function

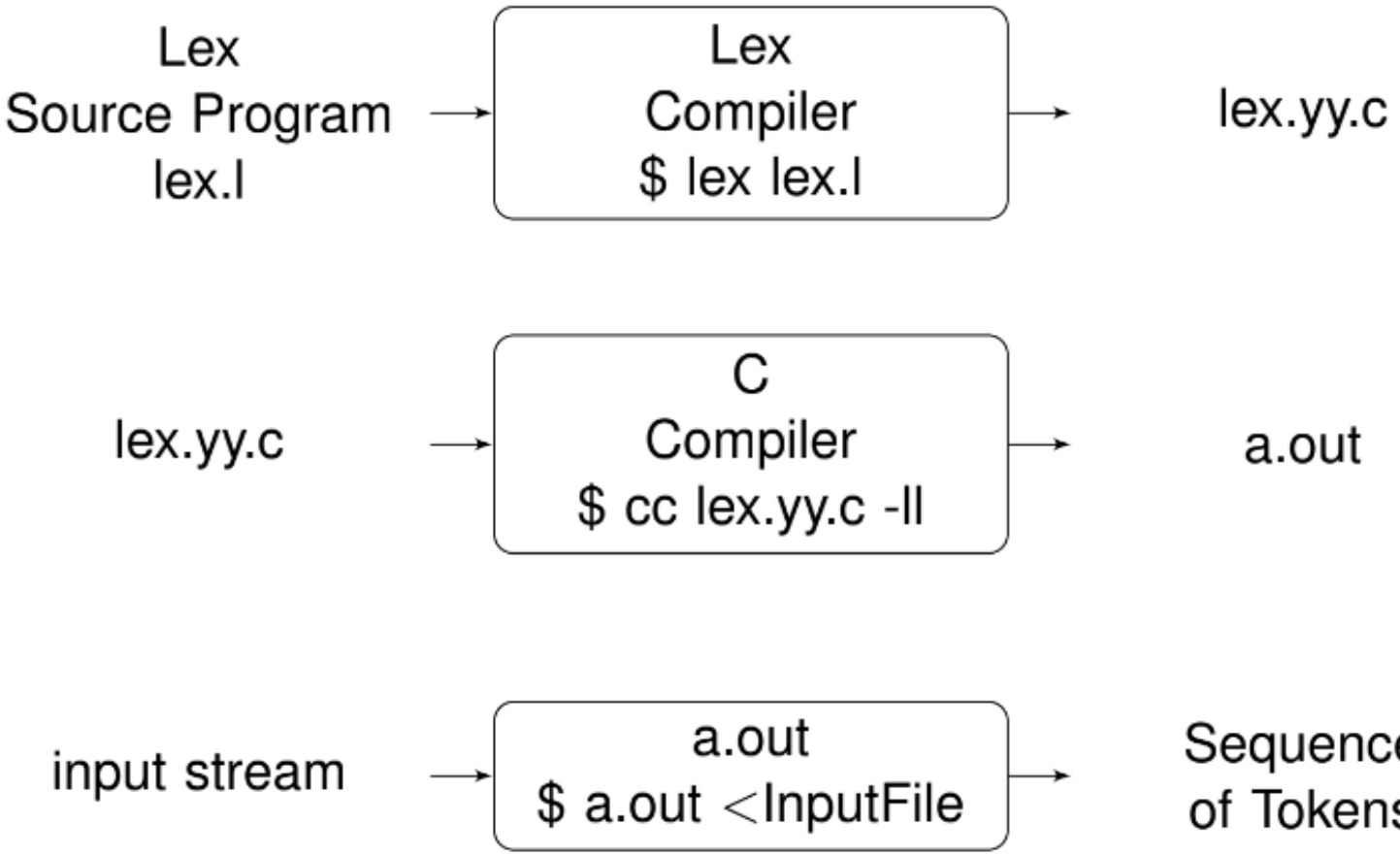
In lex program, a main() function is generally included as:

```
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.

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

## Lexical Analyser Generators — Lex



# Assignment

**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.

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

## tokendef LEX.h

### 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
/*
.
.
.*/
```

IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)

# References


- Alfred V. Aho, Ravi Sethi, Jeffrey D Ullman, “Compilers Principles Techniques and Tools”, Pearson Education.


IIEST, Shibpur
आई आई ई एस टि, शिवपुर
Erstwhile B E College (Estd 1856)