2 Lexical Analysis

Lexical analyzer:

- The lexical analyzer reads the input source program and produces as output a sequence of tokens that the parser uses for syntax analysis.
- The lexical analyzer separates the input into various types of tokens (some meaning full strings) like keywords, Identifiers, special symbols, constants, and operators.
- The part of the input stream that qualifies for a certain type is called a lexeme.
- The lexical analyzer keeps track of newline characters so that it can output the line number with an associated error message.
- The lexical analyzer recognises the comments, white space and stripes out in the source program.
- The lexical analyzer outputs the token that matches the longest possible prefix.
- Lexical analyzer can be implemented with the deterministic finite automata.
- Lexical analyzer, in conjunction with the parser, is responsible for creating symbols to get the next table.
- Lexical analyzer is the first phase of a compiler, which is also known as a scanner.

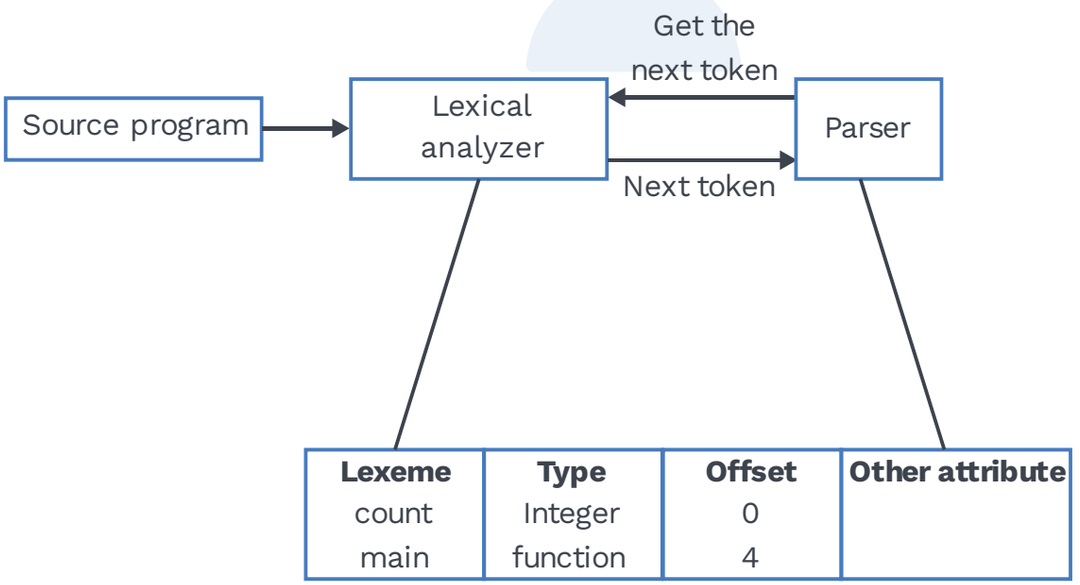


Fig. 2.1 Lexical Analyzer

Keywords:
The table below lists all keywords reserved by the C language.

auto	else	long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while
do	int	struct	_packed
double			

Table 2.1 List of Keywords

Previous Years' Question

In a compiler, keywords of a language are recognised during
a) Parsing of the program
b) The code generation
c) the lexical analysis of the program
d) Dataflow analysis
Sol: c) **[GATE-CS-2011]**

SOLVED EXAMPLES

Q1 **Find the number of tokens in the following C-program.**

```
int main ( )
  {
    int count ;
    /* This is a comment */
    printf ("Hello world\n");
  }
```

Sol: There are 14 tokens in the above C-program.

S.no.	Lexeme	Token
1)	int	keyword
2)	main	Identifier
3)	(	Special symbol
4)	)	Special symbol
5)	{	Special symbol
6)	int	keyword
7)	count	Identifier
8)	;	Separator
9)	printf	Identifier
10)	(	Special symbol
11)	“Hello world\n”	String literal
12)	)	Special symbol
13)	;	Separator
14)	}	Special symbol

Table 2.2 Lexemes to Token

Rack Your Brain

Find the number of tokens in the following C program

```
Int main ( )
  {
    int i = 0;
    while (i < 4)
  {
    printf (“Hi\n”);
  }
    for (i = 0; i < 8; i++);
  {
    printf (“Bye\n”);
  }
  }
```

Previous Years’ Question

The number of tokens in the following C statement is
printf(“i = %d, &i = %x”, i, &i);

a) 3 **b)** 26
c) 10 **d)** 21

Sol: c) **[GATE: CS-2000]**

Lexical specification and lexical analyzer generator:

- The lexical specification for any programming language consists of information about identifying each and every token that is defined for it. (e.g., identifying keywords, operators, Identifiers, string literals).
- The rules for the basic building blocks of a language are called its language specification.
- A lexical analyzer generator is a tool that can generate a code to perform lexical analysis of the input.
- A notation called regular expression is used to write lexical specifications.
- In order to understand the concept of regular expression, we shall use a utility 'egrep' {extended global regular expression print) available on Linux, Unix platform.
- Lex, Flex, JFlex are commonly available tools for lexical analyzer generators to understand the process of generating the lexical analyzer from lexical specifications.

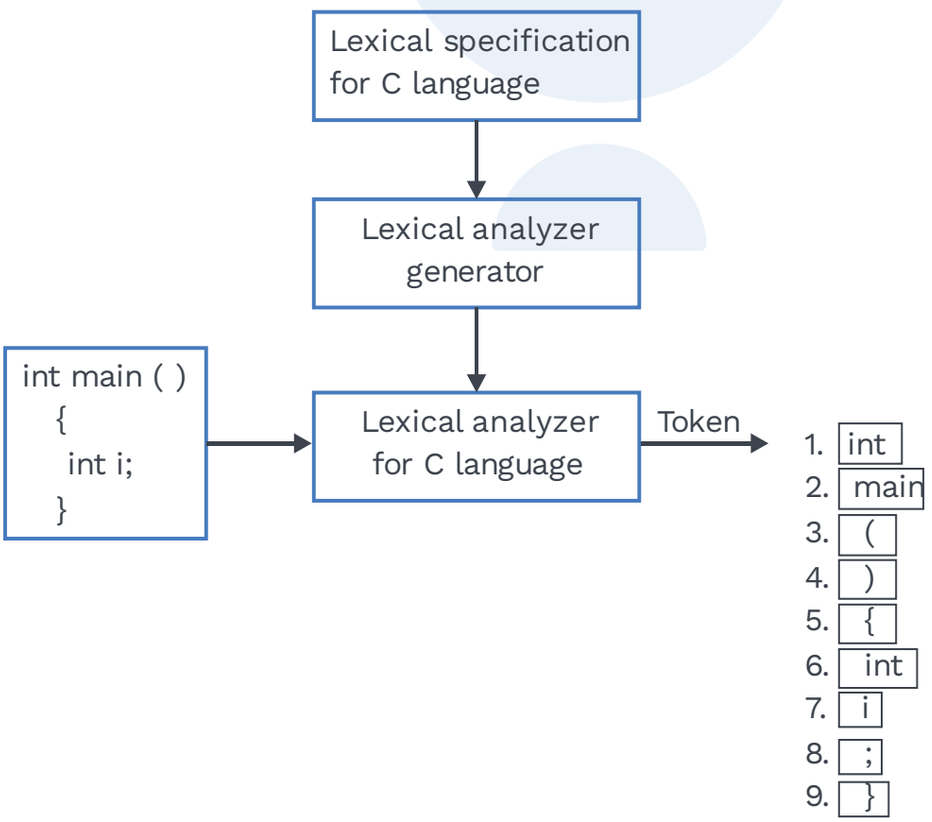
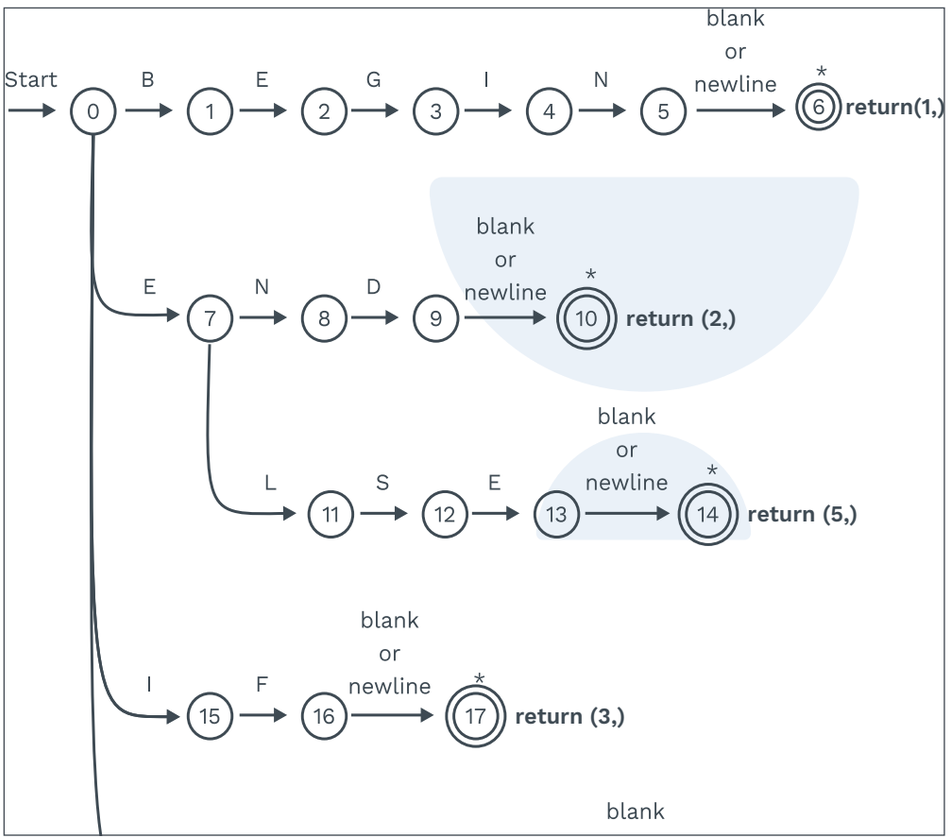


Fig. 2.2 Lexemes to Token Conversion

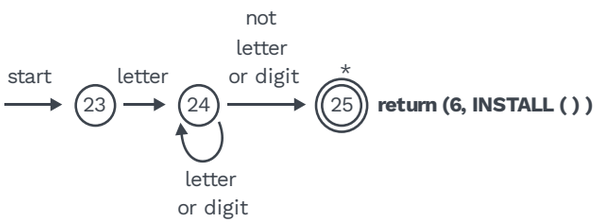
Design of lexical analyzer:

- Designing of any program is to describe the behaviour of the program by a flowchart.
- In lexical analyzer, we use a special kind of flow chart known as transition diagram.

Transition diagram for keywords

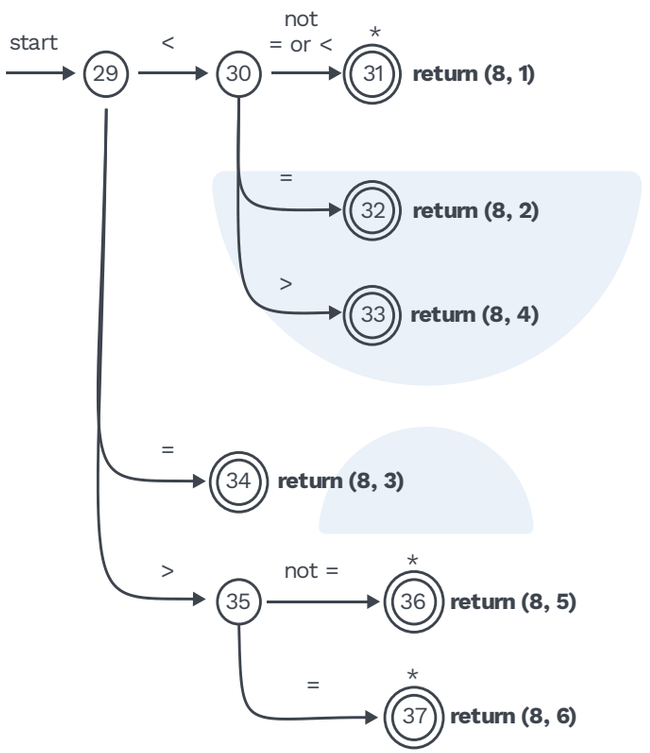


Transition diagram for identifier:



Transition diagram for constant:

Transition diagram for relational operators:



Previous Years' Question

In some programming languages, an identifier is permitted to be later followed by any number of letters or digits. If L and D denote the set of letters and digits, respectively. Which of the following expression defines an identifier?

a) (L + D)*
b) (L.D)*
c) L(L + D)*
d) L(L.D)*

Sol: c)

[ISRO-CS-2017]

Previous Years' Question

A lexical analyzer uses the following patterns to recognise three tokens T1, T2, and T3 over the alphabet {a,b,c}. T1: a?(b|c)*a T2: b?(a|c)*b T3: c?(b|a)*c Note that 'x?' means 0 or 1 occurrence of the symbol x. Note also that the analyzer outputs the token that matches the longest possible prefix. If the string bbaacabc is processed by the analyzer, which one of the following is the sequence of tokens it outputs?

a) $T_1T_2T_3$ **b)** $T_1T_1T_3$
c) $T_2T_1T_3$ **d)** T_3T_3

Sol: d) **[GATE: CS-2018]**

LEX tool:

- LEX tool can produce a lexical analyzer automatically.
- A lex source program contains a set of regular expressions and an action for each expression that is basically used to specify the lexical analyzer.
- When a regular expression is recognised the corresponding action of the piece of code is executed.
- LEX gives output as a lexical analyzer program generated from the source specification.

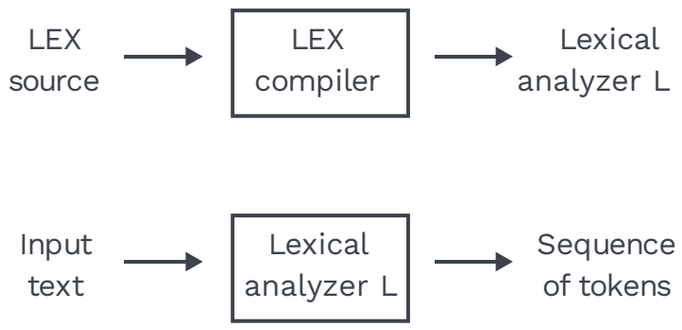
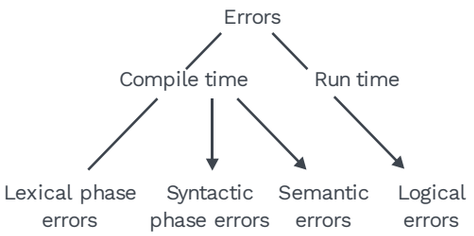


Fig 2.3 The Role of LEX

Error detection in the compiler:

- Compiler detects the errors in the codes made by a programmer and inform, the programmer after compilation. The process of locating an error is known as error detection.
- Error detection and informing the programmer about the error is known as error handling.

Classification of errors:



Lexical phase error:

- Lexical errors are detected during the lexical phase.
- The lexical phase can detect errors when the character remaining in the input does not form any token of the language. Typical lexical errors are:
  1) Appearance of illegal identifiers.
  2) Unmatched strings.
  3) Token in the source program is misspelt.
  4) Exceeding the length of identifiers or numeric constants.

Example:
printf (“prepladder”); $
This is a lexical error since the illegal character $ appears at the end of the statement.

Rack Your Brain

Which of the following gives a lexical error?

a) a123; **b)** 1a;
c) 1 **d)** a_1;

Syntactic phase error:
These errors are detected during syntactic phase Typical syntax errors are:

1) Missing semicolon
2) Unbalanced parentheses
3) Missing operators

Semantic error:
These errors are detected during the semantic analysis phase; typical semantic errors are:

1) Undeclared variables
2) Incompatible type of operands
3) Function computability
4) Type checking.

Logical error:
These errors are detected during runtime. Typical logical errors are:

1) Infinite loop
2) Code not reachable.

Chapter Summary

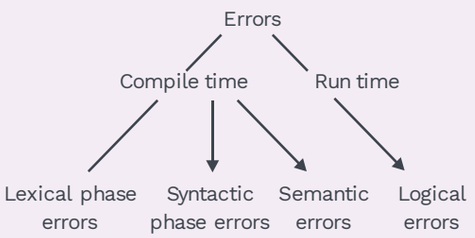
Lexical analyzer:

- The lexical analyzer separates the input into various types of tokens like keywords, identifiers, special symbols and constants operators.
- The lexical analyzer can be implemented with the deterministic finite automata.
- Lex, Flex, JFlex are commonly available tools for lexical analyzer generators to understand the process of generating the lexical analyzer from lexical specifications.

LEX tool:

- A lex source program is a specification of a lexical analyzer, consisting of a set of regular expressions together with an action for each regular expression.

Classification of errors:



Lexical phase error:

1) Appearance of illegal identifiers.
2) Unmatched strings.
3) Token in the source program is misspelt.
4) Exceeding the length of identifiers or numeric constants.

Syntactic phase error:

1) Missing semicolon.
2) Unbalanced parentheses.
3) Missing operators.

Semantic error:

1) Undeclared variables.
2) Incompatible type of operands.
3) Function computability.
4) Type checking.

Logical error:

1) Infinite loop.
2) Code not reachable.