1.5 TOKENS:

A token is a string of characters, categorized according to the rules as a symbol (e.g., IDENTIFIER, NUMBER, COMMA). The process of forming tokens from an input stream of characters is called tokenization.

A token can look like anything that is useful for processing an input text stream or text file. Consider this expression in the C programming language: sum=3+2;

Lexeme	Token type
sum	Identifier
=	Assignment operator
3	Number
+	Addition operator
2	Number
;	End of statement

Pattern:

A pattern is a description of the form that the lexemes of a token may take. A set of strings in the input for which the same token is produced as output. This set of strings is described by a rule called a pattern associated with the token.

Token:

Token is a sequence of characters that can be treated as a single logical entity.

Typical tokens are, 1) Identifiers 2) keywords 3) operators 4) special symbols

5)constants

Lexeme:

Collection or group of characters forming tokens is called Lexeme.

In the case of a keyword as a token, the pattern is just the sequence of characters that form the keyword. For identifiers and some other tokens, the pattern is a more complex structure that is matched by many strings.

Attributes for Tokens

Some tokens have attributes that can be passed back to the parser. The lexical analyzer collects information about tokens into their associated attributes. The attributes influence the translation of tokens.

i) Constant : value of the constant ii)

Identifiers: pointer to the corresponding

symbol table entry.

1.5.1 Error Recovery Strategies in Lexical Analysis:

The following are the error-recovery actions in lexical analysis:

1) Deleting an extraneous character.
2) Inserting a missing character.
3) Replacing an incorrect character by a correct character.
4) Transforming two adjacent characters.
5) **Panic mode recovery**: Deletion of successive characters from the token until error is resolved.

1.5.2 Specification Of Tokens

There are 3 specifications of tokens:

1) Strings
2) Language
3) Regular expression

Strings and Languages

An alphabet or character class is a finite set of symbols.

A string over an alphabet is a finite sequence of symbols drawn from that alphabet.

A language is any countable set of strings over some fixed alphabet.

In language theory, the terms "sentence" and "word" are often used as synonyms for "string." The length of a string s, usually written |s|, is the number of occurrences of symbols in s.For example, banana is a string of length six. The empty string, denoted e, is the string of length zero.

Operations on strings

The following string-related terms are commonly used:

1. A **prefix** of string s is any string obtained by removing zero or more symbols from the end of string s.

For example, ban is a prefix of banana. example, ban is a prefix of banana.

2. A **suffix** of string s is any string obtained by removing zero or more symbols from the beginning of s.

For example, nana is a suffix of banana.

3. A **substring** of s is obtained by deleting any prefix and any suffix from s.

For example, nan is a substring of banana.

4. The **proper prefixes, suffixes**, and **substrings** of a string s are those prefixes, suffixes, and substrings, respectively of s that are not e or not equal to s itself.

5. A subsequence of s is any string formed by deleting zero or more not necessarily consecutive positions of s.

For example, baan is a subsequence of banana.

Operations on languages:

The following are the operations that can be applied to languages:

1. Union
2. Concatenation
3. Kleene closure
4. Positive closure

The following example shows the

operations on strings: Let L={0,1}

and S={a,b,c}

1. Union : L U S={0,1,a,b,c}
2. Concatenation : L.S={0a,1a,0b,1b,0c,1c}
3. Kleene closure : L*={ e,0,1,00….}
4. Positive closure : L+={0,1,00….}