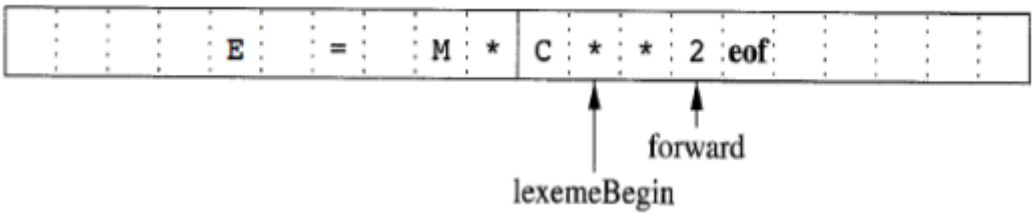
INPUT BUFFERING

- We often have to look one or more characters beyond the next lexeme before we can be sure we have the right lexeme.
- As characters are read from left to right, each character is stored in the buffer to form a meaningful token. We introduce a two-buffer scheme that handles large look a head's safely. We then consider an improvement involving "sentinels" that saves time checking for the ends of buffers.

Buffer Pairs:

- A buffer is divided into two N-character halves, as shown below.
- Each buffer is of the same size N, and N is usually the number of characters on one disk block. E.g., 1024 or 4096 bytes.
- Using one system read command we can read N characters into a buffer.
- If fewer than N characters remain in the input file, then a special character, represented by eof, marks the end of the source file.

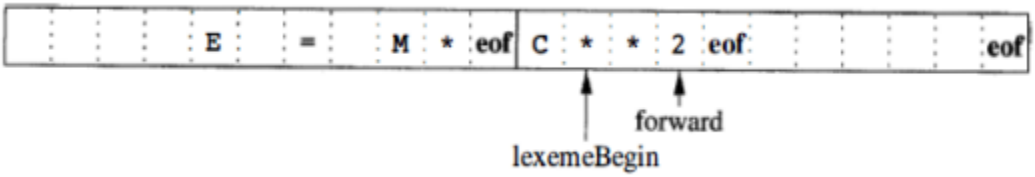


Two pointers to the input are maintained:

- Pointer lexeme_beginning, marks the beginning of the current lexeme, whose extent we are attempting to determine.
- Pointer forward scans ahead until a pattern match is found.

Sentinels:

- For each character read, we make two tests: one for the end of the buffer, and one to determine what character is read. We can combine the buffer-end test with the test for the current character if we extend each buffer to hold a sentinel character at the end.
- The sentinel is a special character that cannot be part of the source program, and a natural choice is the character eof.
- The sentinel arrangement is as shown below:



SPECIFICATION OF TOKEN:

- Regular expressions are notation for specifying patterns.
- Each pattern matches a set of strings.
- Regular expressions will serve as names for sets of strings.
- Strings and Languages String means a finite sequence of symbols.

For example

computer (c, o, m, p, u, t, e, r)

CS6660 (C, S, 6, 6, 6, 0)

101001 (1, 0)

- Symbols are given through alphabet. An alphabet is a finite set of symbols
- The term alphabet or character class denotes any finite set of symbols. e.g., set {0,1} is the binary alphabet.
- The term sentence and word are often used as synonyms for the term string.
- The length of a string s is written as | s | - is the number of occurrences of symbols
- The empty string denoted by ε – length of empty string is zero.
- The term language denotes any set of strings over some fixed alphabet.

Operations on Languages:

There are several operations that can be applied to languages:

Definitions of operations on languages L and M:

OPERATION	DEFINITION
Union of L and M. written LυM	L υ M = { s | s is in L or s is in M }
Concatenation of L and M. written LM	LM = { st | s is in L and t is in M }
Kleene closure of L. written L*	$L^* = \bigcup_{i=0}^{\infty} L^i$ L* denotes "zero or more concatenation of" L.
Positive closure of L. written L+	$L^+ =$ L^+ denotes "one or more Concatenation of" L.

Regular Expressions:

- It allows defining the sets to form tokens.
- Defines a Pascal identifier –identifier is formed by a letter followed by zero or more letters or digits. e.g., letter (letter | digit) *
- A regular expression is formed using a set of defining rules.
- Each regular expression r denotes a language L(r).

Order of evaluate Regular expression:

As defined, regular expressions often contain unnecessary pairs of parentheses. We may drop certain pairs of parentheses if we adopt the conventions that:

- The unary operator * has highest precedence and is left associative.
- Concatenation has second highest precedence and is left associative.
- | has lowest precedence and is left associative.

RECOGNITION OF TOKENS:

We learn how to express pattern using regular expressions. Now, we must study how to take the patterns for all the needed tokens and build a piece of code that examines the input string and finds a prefix that is a lexeme matching one of the patterns.

Grammar for branching statements:

Stmt →if expr then stmt

| If expr then else stmt

| ϵ

Expr →term relop term

| term

Term →id

|number

The terminal of grammar, which are if, then, else, relop,id and numbers are the names of tokens as far as the lexical analyzer is concerned, the patterns for the tokens are described using regular definitions.