SSN COLLEGE OF ENGINEERING, KALAVAKKAM
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Compiler Design – UCS1602
Programming Assignment-2 - Implementation of Lexical Analyzer using LEX tool

Write Lexical Analyzer using the tool LEX. Your scanner will run through the source program recognizing C tokens in the order in which they are read, until end of file is reached. When an identifier is encountered, it should be stored in symbol table with its attributes. Symbol table consist of the attributes identifier name, type, no of bytes and value. **You must implement the scanner for all lexical constructs that are discussed below.**

C Programming Language: Lexical constructs

Here is the summary of the **token types** in C programming language. Refer the regular expressions for the following constructs in assignment 2.

The following are **keywords**. They are all reserved:
auto break case char const continue default do double else enum extern float for goto if int long register return short signed sizeof static struct switch typedef union unsigned void volatile while

An **identifier** is a sequence of letters, digits and underscores, starting with a letter. C language is case sensitive. Eg; "if" is a keyword, but "IF" is an identifier. Binky and binky are two different identifiers.

C language adopts two types of **comments**. A single-line comment is started by **//** and extends to the end of the line. Multi-line comments start with **/*** and end with the first subsequent ***/**. Multi-line comments do not nest. If a file ends with an unterminated comment, the scanner should report an error.

An integer **constant** can either be specified in decimal (base 10) or hexadecimal (base 16). A decimal integer is a sequence of decimal digits (**0-9**). A double constant is a sequence of digits, a period, followed by any sequence of digits. A string constant is a sequence of characters enclosed in double quotes. Strings can contain any character except a new line or double quote. A string must start and end on a single line; it cannot be split over multiple lines:

Arithmetic **operators**

+, -, *, /, %

Arithmetic assignment operators

+=, -=, *=, /=, %=

Logical operators

&&, ||, !

Relational operators

<. <=, >, >=, ==, !=

Bitwise operators

^, &, |, <<, >>

Unary operators

-, ++, - -

Assignment operator

=

Special character

; , . [ ] ( ) { } []

The following are identified as function calls

printf(), scanf(), getch(), clrscr(), getch()

Below is a sample input and output. Test case may be anything comprising of all possible constructs.

Example input source program

```
{
  int a=10,b=20;
  if(a>b)
     printf(“a is greater”);
   else
     printf(“b is greater”);
}
```

Output:

{ - special character
int – keyword
a – identifier
= - assignment operator
10 – integer constant
, -special character
b – identifier
= - assignment operator
20 – integer constant
; - special character
if - keyword
(- special character
a – identifier
> - relational operator
b – identifier
) - special character
printf(“a is greater”) – function call
; - special character
printf(“b is greater”) – function call
; - special character
} - special character