Government College of Engineering, Jalgaon

(An Autonomous Institute of Govt. of Maharashtra)

Name of Student:……
PRN:……. …
Course Teacher: Sharayu Bonde
Date of Performance: **Date of completion:**

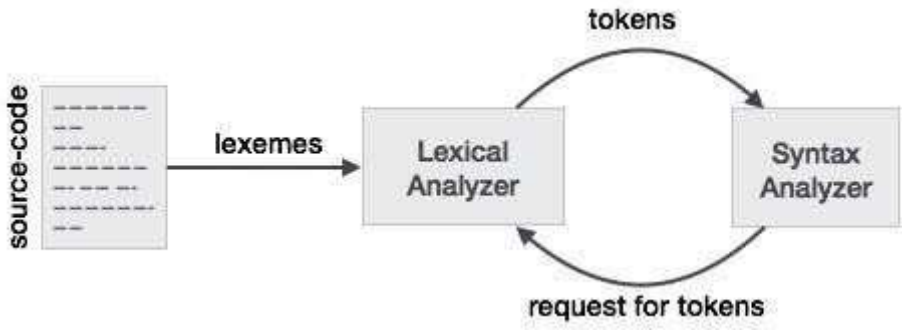
Experiment No.: Group A: 04

Title Write a C program to simulate lexical analyzer for validating operators.

Theory:

Lexical analysis is the first phase of a compiler. It takes the modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

If the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.



C Operators: An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators –

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators

Arithmetic Operators: These are used to perform arithmetic/mathematical operations on operands. The binary operators falling in this category are:

- **Addition:** The **'+'** operator adds two operands. For example, **x+y**.
- **Subtraction:** The **'-'** operator subtracts two operands. For example, **x-y**.
- **Multiplication:** The **'*'** operator multiplies two operands. For example, **x*y**.
- **Division:** The **'/'** operator divides the first operand by the second. For example, **x/y**.
- **Modulus:** The **'%'** operator returns the remainder when first operand is divided by the second. For example, **x%y**.

Relational_Operators:
Relational operators are used for comparison of two values. Let's see them one by one:

- **'=='** operator checks whether the two given operands are equal or not. If so, it returns true. Otherwise it returns false. For example, **5==5** will return true.

- **'!='** operator checks whether the two given operands are equal or not. If not, it returns true. Otherwise it returns false. It is the exact boolean complement of the **'=='** operator. For example, **5!=5** will return false.
- **'>'** operator checks whether the first operand is greater than the second operand. If so, it returns true. Otherwise it returns false. For example, **6>5** will return true.
- **'<'** operator checks whether the first operand is lesser than the second operand. If so, it returns true. Otherwise it returns false. For example, **6<5** will return false.
- **'>='** operator checks whether the first operand is greater than or equal to the second operand. If so, it returns true. Otherwise it returns false. For example, **5>=5** will return true.
- **'<='** operator checks whether the first operand is lesser than or equal to the second operand. If so, it returns true. Otherwise it returns false. For example, **5<=5** will also return true.

Logical_Operators:
They are used to combine two or more conditions/constraints or to complement the evaluation of the original condition in consideration. They are described below:

- **Logical AND:** The **'&&'** operator returns true when both the conditions in consideration are satisfied. Otherwise it returns false. For example, **a && b** returns true when both a and b are true (i.e. non-zero).
- **Logical OR:** The **'||'** operator returns true when one (or both) of the conditions in consideration is satisfied. Otherwise it returns false. For example, **a || b** returns true if one of a or b is true (i.e. non-zero). Of course, it returns true when both a and b are true.
- **Logical NOT:** The **'!'** operator returns true the condition in consideration is not satisfied. Otherwise it returns false. For example, **!a** returns true if a is false, i.e. when a=0.

Bitwise Operators

- In C, following 6 operators are bitwise operators (work at bit-level)
- **& (bitwise AND)** Takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
- **| (bitwise OR)** Takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 any of the two bits is 1.
- **^ (bitwise XOR)** Takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
- **<< (left shift)** Takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
- **>> (right shift)** Takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
- **~ (bitwise NOT)** Takes one number and inverts all bits of it

Assignment Operator :

An Assignment operator is used to form an assignment expression, which assigns the value to an identifier. The most commonly used assignment operator is = .

Result:

Thus the program to stimulate lexical analyzer was executed and the output was verified successfully.

Course Teacher

Sharayu N. Bonde