**Lexical Analyzer Code Explanation**

**Code:**

**#include <stdio.h>**

**#include <ctype.h>**

**int charClass;**

**char lexeme[100];**

**char nextChar;**

**int lexLen;**

**int token;**

**int nextToken;**

**FILE \*in\_fp, \*fopen();**

**#define LETTER 0**

**#define DIGIT 1**

**#define UNKNOWN 99**

**#define INT\_LIT 10**

**#define IDENT 11**

**#define ASSIGN\_OP 20**

**#define ADD\_OP 21**

**#define SUB\_OP 22**

**#define MULT\_OP 23**

**#define DIV\_OP 24**

**#define LEFT\_PAREN 25**

**#define RIGHT\_PAREN 26**

**void addChar() {**

**if (lexLen <= 98) {**

**lexeme[lexLen++] = nextChar;**

**lexeme[lexLen] = 0;**

**} else {**

**printf("Error - lexeme is too long\n");**

**}**

**}**

**void getChar() {**

**if ((nextChar = getc(in\_fp)) != EOF) {**

**if (isalpha(nextChar))**

**charClass = LETTER;**

**else if (isdigit(nextChar))**

**charClass = DIGIT;**

**else**

**charClass = UNKNOWN;**

**} else {**

**charClass = EOF;**

**}**

**}**

**void getNonBlank() {**

**while (isspace(nextChar))**

**getChar();**

**}**

**int lookup(char ch) {**

**switch (ch) {**

**case '(':**

**addChar();**

**nextToken = LEFT\_PAREN;**

**break;**

**case ')':**

**addChar();**

**nextToken = RIGHT\_PAREN;**

**break;**

**case '+':**

**addChar();**

**nextToken = ADD\_OP;**

**break;**

**case '-':**

**addChar();**

**nextToken = SUB\_OP;**

**break;**

**case '\*':**

**addChar();**

**nextToken = MULT\_OP;**

**break;**

**case '/':**

**addChar();**

**nextToken = DIV\_OP;**

**break;**

**default:**

**addChar();**

**nextToken = EOF;**

**break;**

**}**

**return nextToken;**

**}**

**int lex() {**

**lexLen = 0;**

**getNonBlank();**

**switch (charClass) {**

**case LETTER:**

**addChar();**

**getChar();**

**while (charClass == LETTER || charClass == DIGIT) {**

**addChar();**

**getChar();**

**}**

**nextToken = IDENT;**

**break;**

**case DIGIT:**

**addChar();**

**getChar();**

**while (charClass == DIGIT) {**

**addChar();**

**getChar();**

**}**

**nextToken = INT\_LIT;**

**break;**

**case UNKNOWN:**

**lookup(nextChar);**

**getChar();**

**break;**

**case EOF:**

**nextToken = EOF;**

**strcpy(lexeme, "EOF");**

**break;**

**}**

**printf("Next token is: %d, Next lexeme is: %s\n", nextToken, lexeme);**

**return nextToken;**

**}**

**int main() {**

**if ((in\_fp = fopen("front.in", "r")) == NULL) {**

**printf("ERROR - cannot open front.in\n");**

**} else {**

**getChar();**

**do {**

**lex();**

**} while (nextToken != EOF);**

**// Modified arithmetic operation**

**int sum = 100; // Example value for sum**

**int total = 20; // Example value for total**

**float result;**

**// New operation**

**result = (sum - 30) \* total;**

**printf("The result of the modified operation is: %.2f\n", result);**

**}**

**return 0;**

**}**

**\*Imports**

**- #include <stdio.h>:** Used for input and output functions, like printing and working with files.

**- #include <ctype.h>:** Provides functions to check characters .

--------------------------------------------------------------------------------

**Global Variables**

**- charClass:** Stores the type of the current character (letter, digit, or unknown).

**- lexeme[]:** Array where we build tokens like words or numbers.

**- nextChar:** Holds the next character being read.

**- lexLen:** Keeps track of how many characters are in the current token.

- **token and nextToken:** Used to represent and work with tokens in the code.

**- in\_fp:** File pointer for reading the input file.

-------------------------------------------------------------------

\***Defines**

These are constants that make the code easier to read

**- Types of characters:**

**- LETTER (0):** If it's a letter.

**- DIGIT (1):** If it's a number.

**- UNKNOWN (99):** If we don’t recognize it.

- Token Types:

- Numbers (INT\_LIT), words (IDENT), and operators like +, -, \*, /.

-------------------------------------------------------------------------------------------------

**\*Functions**

**1. addChar()**

Adds the current character to the token being built. If it’s too long, prints an error.

**2. getChar()**

Reads the next character from the file. Decides if it’s a letter, number, or unknown.

**3. getNonBlank()**

Skips all blank spaces and goes straight to the next non-whitespace character.

**4. lookup(ch)**

Checks if the character is an operator (+, -, \*, /) or a parenthesis ((, )).

**5. lex()**

Reads characters, classifies them, and builds tokens like words or numbers. Prints the token type and value.

-------------------------------------------------------------------------------

**\* Main Function**

**1. Opens the file (front.in):**

- If it can’t open, prints an error.

- If opened successfully, starts reading the file.

**2. Token Analysis:**

- Processes the file and identifies tokens until it reaches the end (EOF).

**3. Arithmetic Operation:**

- Calculates (sum - 30) \* total using two numbers and prints the result.

-----------------------------------------------------------------------------------------

**\* Output Example**

**Next token is:** 10, Next lexeme is: 123

**Next token is:** 21, Next lexeme is: +

**Next token is:** 11, Next lexeme is: variable

The result of the modified operation is: 1400.00