

Analisador léxico em C referente ao diagrama de estados da figura.

```
/* front.c - um sistema analisador léxico para
   expressões aritméticas simples */

#include <stdio.h>
#include <ctype.h>

/* Declarações globais */
/* Variáveis */
int charClass;
char lexeme [100];
char nextChar;
int lexLen;
int token;
int nextToken;

FILE *in_fp, *fopen();
/* Declarações de função */
void addChar();
void getChar();
void getNonBlank();
int lex();

/* Classes de caracteres */
#define LETTER 0
#define DIGIT 1
#define UNKNOWN 99

/* Códigos de tokens */
#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
/*****************/
/* função principal */
main() {

    /* Abre o arquivo de dados de entrada e processa seu conteúdo */
    if ((in_fp = fopen("front.in", "r")) == NULL)
        printf("ERROR - cannot open front.in \n");
    else {
        getChar();
        do {
            lex();
        } while (nextToken != EOF);
    }
}
```

```

/*****************/
/* lookup - uma função para processar operadores e parênteses
   e retornar o token */
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;
}

/*****************/
/* addChar - uma função para adicionar nextChar a lexeme */
void addChar() {
    if (lexLen <= 98) {
        lexeme[lexLen++] = nextChar;
        lexeme[lexLen] = 0;
    }
    else
        printf("Error - lexeme is too long \n");
}

/*****************/
/* getChar - uma função para obter o próximo caractere de
   entrada e determinar sua classe */
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;
}

```

```

/* getNonBlank - uma função para chamar getChar até que ele
   retorne um caractere que não seja um espaço em
   branco */
void getNonBlank() {
    while (isspace(nextChar))
        getChar();
}
/

*****/* lex - um analisador léxico simples para expressões
       aritméticas */

int lex() {
    lexLen = 0;
    getNonBlank();
    switch (charClass) {
/* Analisa identificadores sintaticamente */
    case LETTER:
        addChar();
        getChar();
        while (charClass == LETTER || charClass == DIGIT) {
            addChar();
            getChar();
        }
        nextToken = IDENT;
        break;

/* Analisa literais sintaticamente */
    case DIGIT:
        addChar();
        getChar();
        while (charClass == DIGIT) {
            addChar();
            getChar();
        }
        nextToken = INT_LIT;
        break;

/* Parênteses e operadores */
    case UNKNOWN:
        lookup(nextChar);
        getChar();
        break;

/* Fim do arquivo */
    case EOF:
        nextToken = EOF;
        lexeme[0] = 'E';
        lexeme[1] = 'O';
        lexeme[2] = 'F';
        lexeme[3] = 0;
        break;
    } /* Fim do switch */
    printf("Next token is: %d, Next lexeme is %s\n",
           nextToken, lexeme);
    return nextToken;
} /* Fim da função lex */

```

Entrada:

(sum + 47) / total

A seguir, temos a saída do analisador léxico de front.c quando usado com essa expressão:

Next token is: 25 Next lexeme is (

Next token is: 11 Next lexeme is sum

Next token is: 21 Next lexeme is +

Next token is: 10 Next lexeme is 47

Next token is: 26 Next lexeme is)

Next token is: 24 Next lexeme is /

Next token is: 11 Next lexeme is total

Next token is: -1

Next lexeme is EOF

