Aim: - Write a program to implement simple lexical analyzer using c language.

#include <stdio.h>

#include <ctype.h>

#include <string.h>

// Define the token types

typedef enum {

    IDENTIFIER,

    NUMBER,

    OPERATOR,

    PUNCTUATION,

    UNKNOWN

} TokenType;

// Define the structure of a token

typedef struct {

    TokenType type;

    char value[100];

} Token;

// Function to classify and tokenize the input string

int tokenize(const char\* input, Token tokens[], int maxTokens) {

    int i = 0, tokenIndex = 0;

    while (input[i] != '\0' && tokenIndex < maxTokens) {

        if (isalpha(input[i])) {

            // Process identifier

            int j = 0;

            while (isalnum(input[i]) || input[i] == '\_') {

                tokens[tokenIndex].value[j++] = input[i++];

            }

            tokens[tokenIndex].value[j] = '\0';

            tokens[tokenIndex].type = IDENTIFIER;

            tokenIndex++;

        } else if (isdigit(input[i])) {

            // Process number

            int j = 0;

            while (isdigit(input[i])) {

                tokens[tokenIndex].value[j++] = input[i++];

            }

            tokens[tokenIndex].value[j] = '\0';

            tokens[tokenIndex].type = NUMBER;

            tokenIndex++;

        } else if (ispunct(input[i])) {

            // Process operator or punctuation

            tokens[tokenIndex].value[0] = input[i];

            tokens[tokenIndex].value[1] = '\0';

            if (input[i] == '=' || input[i] == '+' || input[i] == '-' ||

                input[i] == '\*' || input[i] == '/' || input[i] == ';') {

                tokens[tokenIndex].type = OPERATOR;

            } else {

                tokens[tokenIndex].type = PUNCTUATION;

            }

            tokenIndex++;

            i++;

        } else {

            // Process unknown token

            tokens[tokenIndex].value[0] = input[i];

            tokens[tokenIndex].value[1] = '\0';

            tokens[tokenIndex].type = UNKNOWN;

            tokenIndex++;

            i++;

        }

    }

    return tokenIndex; // Return the total number of tokens generated

}

// Function to print the token details

void printTokens(Token tokens[], int tokenCount) {

    for (int i = 0; i < tokenCount; i++) {

        printf("Token Type: ");

        switch (tokens[i].type) {

            case IDENTIFIER: printf("IDENTIFIER"); break;

            case NUMBER: printf("NUMBER"); break;

            case OPERATOR: printf("OPERATOR"); break;

            case PUNCTUATION: printf("PUNCTUATION"); break;

            case UNKNOWN: printf("UNKNOWN"); break;

        }

        printf(", Value: %s\n", tokens[i].value);

    }

}

int main() {

    char input[100];

    Token tokens[100];  // Array to store tokens

    int tokenCount;

    // Read input from the user

    printf("Enter a string to tokenize: ");

    fgets(input, sizeof(input), stdin);

    // Remove the newline character at the end (if any)

    input[strcspn(input, "\n")] = '\0';

    // Tokenize the input string

    tokenCount = tokenize(input, tokens, 100);

    // Print the resulting tokens

    printTokens(tokens, tokenCount);

    return 0;

}

Output: -





