**Name- Arzoo Saini**

**Sap id- 500097324**

**Roll number- R2142211322**

**Course - B.tech CSE (Blockchain-Non honors) 3rd year**

**Instructor - Mr. Saurabh Shanu**

**Github Profile** - <https://github.com/Aarzoo04/MyLang>

**TOKENIZATION**

LEXER CODE:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <stdbool.h>

typedef enum

{

    IDENTIFIER=1,

    CONSTANT=2,

    OPERATOR=3,

    PUNCTUATION=4,

    KEYWORD=5,

    SPECIAL\_SYMBOL=6

} TokenType;

typedef struct

{

    TokenType type;

    char value[50];

} Token;

bool isOperator(char c)

{

    char operators[] = "+-\*/=<>^";

    for (int i = 0; i < strlen(operators); i++)

    {

        if (c == operators[i])

            return true;

    }

    return false;

}

bool isPunctuation(char c)

{

    char punctuations[] = ".,;(){}[]$";

    for (int i = 0; i < strlen(punctuations); i++)

    {

        if (c == punctuations[i])

            return true;

    }

    return false;

}

bool isSpecialSymbol(char c)

{

    char SpecialSymbol[] = "@#&%?";

    for (int i = 0; i < strlen(SpecialSymbol); i++)

    {

        if (c == SpecialSymbol[i])

            return true;

    }

    return false;

}

bool isKeyword(char \*str)

{

    char \*keywords[] = {"if", "else", "while", "for", "int", "decimal", "void", "return", "string", "input", "is", "or", "and", "not", "in", "goto", "switch", "case"};

    int numKeywords = sizeof(keywords) / sizeof(keywords[0]);

    for (int i = 0; i < numKeywords; i++)

    {

        if (strcmp(str, keywords[i]) == 0)

            return true;

    }

    return false;

}

Token \*tokenize(char \*input)

{

    Token \*tokens = malloc(strlen(input) \* sizeof(Token));

    int tokenCount = 0;

    int i = 0;

    while (input[i] != '\0')

    {

        if (isspace(input[i]))

        {

            i++;

            continue;

        }

        if (isalpha(input[i]))

        {

            int j = 0;

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

            {

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

            }

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

            if (isKeyword(tokens[tokenCount].value))

            {

                tokens[tokenCount].type = KEYWORD;

            }

            else

            {

                tokens[tokenCount].type = IDENTIFIER;

            }

            tokenCount++;

            continue;

        }

        if (isdigit(input[i]))

        {

            int j = 0;

            while (isdigit(input[i]) || input[i] == '.')

            {

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

            }

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

            tokens[tokenCount].type = CONSTANT;

            tokenCount++;

            continue;

        }

        if (isOperator(input[i]))

        {

            tokens[tokenCount].value[0] = input[i++];

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

            tokens[tokenCount].type = OPERATOR;

            tokenCount++;

            continue;

        }

        if (isPunctuation(input[i]))

        {

            tokens[tokenCount].value[0] = input[i++];

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

            tokens[tokenCount].type = PUNCTUATION;

            tokenCount++;

            continue;

        }

        if (isSpecialSymbol(input[i]))

        {

            tokens[tokenCount].value[0] = input[i++];

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

            tokens[tokenCount].type = SPECIAL\_SYMBOL;

            tokenCount++;

            continue;

        }

        i++;

    }

    tokens[tokenCount].type = -1;

    return tokens;

}

const char \*tokenTypeNames[] = {"Undefined", "Identifier", "Constant", "Operator", "Punctuation", "Keyword", "Special Symbol"};

void printTokens(Token \*tokens)

{

    int i = 0;

    printf("Value      |      Type\n");

    printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

    while (tokens[i].type != -1)

    {

        printf("%-10s | %-12s\n", tokens[i].value,tokenTypeNames[tokens[i].type]);

        i++;

    }

}

int main()

{

    FILE \*file;

    file = fopen("function4.c", "r");

    int length;

    if (file == NULL)

    {

        return 1;

    }

    fseek(file, 0, SEEK\_END);

    length = ftell(file);

    fseek(file, 0, SEEK\_SET);

    char input[length + 1];

    size\_t items\_read = fread(input, 1, length, file);

    input[length] = '\0';

    fclose(file);

    Token \*tokens = tokenize(input);

    printTokens(tokens);

    free(tokens);

    return 0;

}

OUTPUTS of Different Functions:















