Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Course No. : CSE4130
Course Name : Formal Languages and Compilers Lab

Assignment No. : 02

Submitted By:

Name : Shahriar Hasan Chowdhury
ID No. : 17 01 04 030
Session : Spring - 2020
Section : A (A2)

OBJECTIVES:

To write a program that reads any simple program as source and separates out the valid tokens from the source program.

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


int isSeperator(char c)
{
    if(c == ';' || c == '\''  || c =='\"' || c == ',')
        return 1;
    return 0;
}

int isOperator(char c)
{
    if(c == '+' || c == '-' || c == '*' || c == '/' ||
        c == '=' || c == '>' || c == '<' || c == '!' ||
         c == '%' || c == '&' || c == '^' || c == '~')
         return 1;
    return 0;
}

int isDoubleOperator(char c1, char c2)
{
    if((c1 == '+' && c2 == '+') || (c1 == '-' && c2 == '-
') || (c1 == '+' && c2 == '=') ||
        (c1 == '-
' && c2 == '=') || (c1 == '=' && c2 == '=') || (c1 == '>' && c2 == '=') ||
        (c1 == '<' && c2 == '=') || (c1 == '&' && c2 == '&') || (c1 == '|' &&
 c2 == '|') ||
        (c1 == '>' && c2 == '>') || (c1 == '<' && c2 == '<') || (c1 == '?' &&
 c2 == ':'))
            return 1;
    return 0;
}

int isParenthesis(char c)
{
    if(c == '(' || c == ')' || c == '{' || c == '}' || c == '[' || c == ']')
        return 1;
    return 0;
}

char keywords[32][32] =
{
```

```
    "auto", "const", "double", "float", "int", "short", "struct", "unsigned",

    "break", "continue", "else", "for", "long", "signed", "switch", "void",
    "case", "default", "enum", "goto", "register", "sizeof", "typedef", "vola
tile",
    "char", "do", "extern", "if", "return", "static", "union", "while"
};

int isKeyword(char arr[])
{
    for(int i=0; i<32; i++)
    {
        if(strcmp(keywords[i], arr) == 0)
            return 1;
    }
    return 0;
}

int isIdentifier(char arr[])
{
    if (!(isalpha(arr[0]) || arr[0]== '_'))
       return 0;

    for (int i = 1; i < strlen(arr); i++)
    {
        if (!(isalpha(arr[i]) || arr[i] == '_' || isdigit(arr[i])))
            return 0;
    }

    return 1;
}


int isRealNumber(char arr[])
{
    int NumOfPoint = 0;
    int digit = 1;
    for(int i = 0; i < strlen(arr); i++)
    {
        if(isdigit(arr[i]))
            digit = 1;
        else if(arr[i] == '.')
            NumOfPoint++;
        else
        {
            digit = 0;
            break;
        }
    }
```

```
    if(arr[strlen(arr)-1] == '.')
        return 0;

    if(digit == 1 && NumOfPoint <= 1)
        return 1;
    return 0;
}

void categorizeLexemes(char input2[])
{
    int endIndex = 0;
    char id[100];
    for(int i=0;i<strlen(input2);i++)
    {
        if(isParenthesis(input2[i]))
        {
            printf("[par %c] ", input2[i]);
        }
        else if(isSeperator(input2[i]))
        {
            printf("[sep %c] ",input2[i]);
        }
        else if(isOperator(input2[i]))
        {
            if(isOperator(input2[i+1]))
            {
                printf("[op %c%c] ", input2[i++], input2[i]);
            }
            else
                printf("[op %c] ",input2[i]);
        }
        else
        {
             if(input2[i] != ' ')
             {
                 id[endIndex] = input2[i];
                 endIndex++;
             }
             else
             {
                 id[endIndex] = '\0';
                 endIndex = 0;
                 if(strlen(id) >= 1)
                 {
                     if(isKeyword(id))
                         printf("[kw %s] ",id);
                     else if(isIdentifier(id))
                         printf("[id %s] ",id);
                     else if(isRealNumber(id))
                         printf("[num %s] ",id);
```

```
                    else
                        printf("[unkn %s] ",id);
                }
            }
        }
    }
}

void separateLexemes(char input[])
{
    FILE *f = fopen("step2input.txt", "w");
    int l = strlen(input);
    for(int i=0;i<l;i++)
    {
        if(isSeperator(input[i]))
        {

            fputc(' ', f);
            fputc(input[i], f);
            fputc(' ', f);
        }
        else if(isParenthesis(input[i]))
        {

            fputc(' ', f);
            fputc(input[i], f);
            fputc(' ', f);

        }
        else if(isDoubleOperator(input[i], input[i+1]))
        {
            fputc(' ', f);
            fputc(input[i], f);
            i++;
            fputc(input[i], f);
            fputc(' ', f);
        }
        else if(isOperator(input[i]))
        {
            fputc(' ', f);
            fputc(input[i], f);
            fputc(' ', f);
        }
        else{
            fputc(input[i], f);
        }
    }
    fclose(f);

}
```

```c
int main()
{
    char input[100000];
    char input2[100000];

    FILE *f1 = fopen("input.txt", "r");
    if(f1) fgets(input, 100000, f1);
    else printf("File Not Found");
    fclose(f1);

    separateLexemes(input);

    FILE *f2 = fopen("step2input.txt", "r");
    if(f2) fgets(input2, 100000, f2);
    else printf("File Not Found");
    fclose(f2);

    // printf("\n============================\n");
    // for(int i=0;i<strlen(input2);i++)
    //     printf("%c",input2[i]);
    // printf("\n============================\n");

    categorizeLexemes(input2);
    return 0;

}
```