**SSN COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE**

**UCS1602 – COMPILER DESIGN**

DATE: 4-02-2021

NAME: KEERTHANA T

REGISTER NUMBER: 185001074

CLASS AND SEC : CSE-B

**ASSIGNMENT -1 : LEXICAL ANALYSER**

**FILES:**

* Ex1.c
* Program.txt

**CODE:**

Ex1.c

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

int main()

{

    char ch, buffer[1000];

    FILE \*fp;

    int i,j=0;

    char keywords[32][10] = {"auto","break","case","char","const","continue","default",

                            "do","double","else","enum","extern","float","for","goto",

                            "if","int","long","register","return","short","signed",

                            "sizeof","static","struct","switch","typedef","union",

                            "unsigned","void","volatile","while"};

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

    char unaryOperators[2][5]={"++","--"};

    char arithmeticOperators[5][5]={"+","-","\*","/","%"};

    char arithmeticAssignmentOperators[5][5]={"+=","-=","\*=","/=","%="};

    char relationalOperators[6][5]={"<=",">=","==","!=",">","<"};

    char logicalOperators[3][5]={"&&","||","!"};

    char bitwiseOperators[5][5]={"^","&","|","<<",">>"};

    fp = fopen("program.txt","r");

    char type[2][10];

    char val[2][10];

    char id[2][10];

    char t[10];

    char v[10];

    char ir[10];

    int l=0;

    if(fp == NULL)

    {

        printf("error while opening the file\n");

        exit(0);

    }

     int k=0;

     while((ch = fgetc(fp)) != EOF)

     {

        buffer[k++]=ch;

     }

     buffer[k]='\0';

     j=0;

     k=0;

     char buff[30];

     while(buffer[j]!='\0')

     {

        if(buffer[j]=='#')

        {

            while(buffer[j]!='\n')

            {

                buff[k++]=buffer[j];

                j++;

            }

            buff[k]='\0';

            printf("\n%s\t->\tpre processor directive",buff);

            k=0;

            memset(buff,0,strlen(buff));

            j++;

        }

        char ch=buffer[j];

        if(isalnum(ch))

        {

            buff[k++]=ch;

        }

        else if(ch==' ' || ch=='\n')

        {

            buff[k]='\0';

            int flag=0,nf=0;

            for(i=0;i<32;i++)

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

                {

                    printf("\n%s\t->\tkeyword",buff);

                    flag=1;

                    memset(t,0,strlen(t));

                    strcpy(t,buff);

                }

                if(flag==0 && buff[0]!='\0')

                {

                    int p=0;

                    while(buff[p]!='\0')

                    {

                        if(isdigit(buff[p])==0)

                        {

                            nf=1;

                        }

                        p++;

                    }

                    if(nf==0)

                    {

                        printf("\n%s\t->\tinteger constant",buff);

                        memset(v,0,strlen(v));

                        strcpy(v,buff);

                        strcpy(type[l],t);

                        strcpy(id[l],ir);

                        strcpy(val[l],v);

                        l++;

                    }

                    else

                    {

                        printf("\n%s\t->\tidentifier",buff);

                    }

                    k=0;

                    memset(buff,0,strlen(buff));

                }

                k=0;

                memset(buff,0,strlen(buff));

        }

        else if(ch=='(')

        {

            buff[k]='\0';

            for(i=0;i<32;i++)

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

                {

                    printf("\n%s\t->\tkeyword",buff);

                    memset(t,0,strlen(t));

                    strcpy(t,buff);

                }

            if(strcmp(buff,"main")==0)

            {

                printf("\nmain()\t->\tfuntion call");

                j=j+2;

                ch='\0';

            }

            if(strcmp(buff,"printf")==0)

            {

                while(buffer[j]!=';')

                    {

                        buff[k++]=buffer[j];

                        j++;

                    }

                buff[k]='\0';

                printf("\n%s\t->\tfuntion call",buff);

            }

            k=0;

            memset(buff,0,strlen(buff));

            ch=buffer[j];

        }

        for(i=0;i<strlen(specialCharacters);i++)

        {

            if(ch==specialCharacters[i])

            {

                printf("\n%c\t->\tspecial character",ch);

            }

        }

        ch=buffer[j];

        if(isalnum(ch)==0 && ch!=' ' && ch!='\n' && ch!='(' && ch!=')' && ch!='.' && ch!=';' && ch!='{' && ch!='}' )

        {

            buff[k]='\0';

            int nf=0;

            if(buff[0]!='\0')

            {

                int p=0;

                while(buff[p]!='\0')

                {

                    if(isdigit(buff[p])==0)

                    {

                        nf=1;

                    }

                    p++;

                }

                if(nf==0)

                {

                    printf("\n%s\t->\tinteger constant",buff);

                    memset(v,0,strlen(v));

                    strcpy(v,buff);

                    strcpy(type[l],t);

                    strcpy(id[l],ir);

                    strcpy(val[l],v);

                    l++;

                }

                else

                {

                    printf("\n%s\t->\tidentifier",buff);

                    memset(ir,0,strlen(ir));

                    strcpy(ir,buff);

                }

            }

            k=0;

            memset(buff,0,strlen(buff));

            while(isalnum(buffer[j])==0)

            {

                buff[k++]=buffer[j++];

            }

            if(strcmp(buff,"=")==0)

            {

                printf("\n%s\t->\tassignment operator",buff);

            }

            for(i=0;i<2;i++)

                if(strcmp(buff,unaryOperators[i])==0)

                {

                    printf("\n%s\t->\tunary operator",buff);

                }

            for(i=0;i<5;i++)

                if(strcmp(buff,arithmeticOperators[i])==0)

                {

                    printf("\n%s\t->\tarithmetic operator",buff);

                }

            for(i=0;i<5;i++)

                if(strcmp(buff,arithmeticAssignmentOperators[i])==0)

                {

                    printf("\n%s\t->\tarithmetic assignment operator",buff);

                }

            for(i=0;i<6;i++)

            {

                if(strcmp(buff,relationalOperators[i])==0)

                {

                    printf("\n%s\t->\trelational operator",buff);

                }

            }

            for(i=0;i<2;i++)

                if(strcmp(buff,unaryOperators[i])==0)

                {

                    printf("\n%s\t->\tunary operator",buff);

                }

            for(i=0;i<3;i++)

                if(strcmp(buff,logicalOperators[i])==0)

                {

                    printf("\n%s\t->\tlogical operator",buff);

                }

            for(i=0;i<5;i++)

                if(strcmp(buff,bitwiseOperators[i])==0)

                {

                    printf("\n%s\t->\tbitwise operator",buff);

                }

            k=0;

            memset(buff,0,strlen(buff));

            j--;

        }

        j++;

     }

    fclose(fp);

    char data\_t[4][10] = {"int","char","short","long"};

    int arr[4]={2,1,2,4};

    int address=1000;

    printf("\n\nSymbol Table:\n");

    printf("\nIDENTIFIER NAME   TYPE    NO.OF.BYTES     ADDRESS     VALUE");

    for(i=0;i<l;i++)

    {

        printf("\n%s\t\t%s\t",id[i],type[i]);

        for(j=0;j<4;j++)

        {

            if(strcmp(data\_t[j],type[i])==0)

            {

                printf("%d\t\t\t%d\t\t%s",arr[j],address,val[i]);

                address+=arr[j];

            }

        }

    }

    return 0;

}

Program.txt

#include<stdio.h>

main()

{

long d =100000;

short s=1;

int f=345;

char a= 'w';

printf(s);

}

**OUTPUT:**

