|  |  |  |
| --- | --- | --- |
| **Compiler Design Laboratory**  |  | | --- | | **Course No:** CSE 3212 | |  |   Project Name: Implementation of Lexical Analyzer |

|  |
| --- |
| **Submitted By:** Ishrak islam zarif  ROLL : 1307025 |

**Problem Statement:**

In this lab, we were assigned to write a program to design a lexical analyzer of our own that will include the followings:

1. Design ID’s of our own
2. Design Key words
3. Design operators
4. Comments
5. Punctuation

We were also assigned to write the lex program to recognize our ID, Keyword, Operator and comments using a source file containing our designed lexeme.

1. IDs Design:

The regular expression for IDs is: [A-Za-z]+[\_]?[0-9]\*

1. Keyword Design:

The regular expression for keywords is: number|print|method|loop|if|elif|else

1. Operators Design:

The regular expression for Operators is: assign|<=|>=|==|!=|[+]|[-]|[\*]|[/]|[>]|[<]

4) Comments:

The regular expression for comments is: [-][-].\*

5) Punctuation:

The regular expression for punctuation is: [`]|[:]|[;]

**Lex Program:**

%{

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

char s[100];

char key\_store[50][50];

int key\_counter=0;

int key\_fetched\_flag=0;

char identification\_store[50][50];

int identification\_counter=0;

int identification\_fetched\_flag=0;

char operator\_store[7][7];

int operator\_counter=0;

int operator\_fetech\_flag=0;

char punctuation\_store[7][7];

int punctuation\_counter=0;

int punctuation\_fetech\_flag=0;

int stm=0;

%}

%%

number|print|method|loop|if|elif|else {

key\_fetched\_flag++;

int i=0;

strcpy(s, yytext);

if(key\_counter>0){

for (i=0; i<key\_counter; i++){

if(!strcmp(key\_store[i], s))

break;

}

}

if(i==key\_counter)

strcpy(key\_store[key\_counter++], s);

}

MyHeader1|MyHeader2 {printf("Library found %s.\n", yytext);}

[(].\*[)] {printf("Function found\n");}

assign|<=|>=|==|!=|[+]|[-]|[\*]|[/]|[>]|[<] {

operator\_fetech\_flag++;

int i=0;

strcpy(s, yytext);

if(operator\_counter>0){

for(i=0; i<operator\_counter; i++){

if(!strcmp(operator\_store[i], s))

break;

}

}

if(i==operator\_counter)

strcpy(operator\_store[operator\_counter++], s);

}

[`]|[:]|[;] {

punctuation\_fetech\_flag++;

int i=0;

strcpy(s, yytext);

if(punctuation\_counter>0){

for(i=0; i<punctuation\_counter; i++){

if(!strcmp(punctuation\_store[i], s))

break;

}

}

if(i==punctuation\_counter)

strcpy(punctuation\_store[punctuation\_counter++], s);

}

[A-Za-z]+[\_]?[0-9]\* {

identification\_fetched\_flag++;

int i=0;

strcpy(s, yytext);

if(identification\_counter>0)

{

for (i=0;i<identification\_counter;i++)

if(!strcmp(identification\_store[i], s))

break;

}

if(i==identification\_counter)

strcpy(identification\_store[identification\_counter++], s);

}

"}" {

int i;

printf("\n\n--------------------------------------------------------\n");

printf("\nKeywords: %d\n", key\_counter);

for (i=0; i<key\_counter; i++){

printf("%s\n",key\_store[i]);

}

printf("\n");

printf("Variables: %d\n", identification\_counter);

for (i=0;i<identification\_counter;i++){

printf("%s\n",identification\_store[i]);

}

printf("\n");

printf("Operators: %d\n", operator\_counter);

for (i=0;i<operator\_counter;i++){

printf("%s\n",operator\_store[i]);

}

printf("\n");

printf("Punctuations: %d\n", punctuation\_counter);

for (i=0;i<punctuation\_counter;i++){

printf("%s\n",punctuation\_store[i]);

}

}

["].\*["] printf("");

[ ]\* printf("");

[-][-].\* printf("single line comment found: %s\n", yytext);

"/\*"(.\*|\n)\*"\*/" printf("");

. printf("");

\n printf("");

%%

int yywrap(){

return 1;

}

int main(){

yyin=freopen("input.c", "r", stdin);

yyout=freopen("output.txt", "w", stdout);

yylex();

return 0;

}

**Input:**

--header start

~MyHeader1

~MyHeader2

--header end

--main function

number method()

{

number x,y,z,i`

number sum`

x assign 1`

y assign 2`

i assign 0`

loop : i <= 5

z assign x + y`

;

sum assign z`

print(sum)`

if : x==1

print(x)`

;

elif : x!=1

print(y)`

;

else

print(i)`

;

}

**Output:**

single line comment found: --header start

Library found MyHeader1.

Library found MyHeader2.

single line comment found: --header end

single line comment found: --main function

Function found

Function found

Function found

Function found

Function found

--------------------------------------------------------

Keywords: 7

number

method

loop

print

if

elif

else

Variables: 5

x

y

z

i

sum

Operators: 5

assign

<=

+

==

!=

Punctuations: 3

`

:

;

**Conclusion:**

In this lab I have learned about the lexical analyzer. I have also learned how it works. I have also implemented it to find the type, identifier, operators, and reserve words. At the time of implementing this lexical analyzer I have faced a lot of problem and errors. By debugging the program, I have some removed some of the errors. I have tried my level best to make my program error fee. From this assignment I have known about the compiler design and top down parsing technique. I have also implemented the top down parsing technique in C language. At the time of implementation, I have faced some problems and solved these problems. At the end I became successful to implement the grammar using top down parsing technique.