Aarav Raj Singh CCIS 01 2015011 Compiler design

**PROGRAM-1**

**Design a LEX Code to count the number of lines, space, tab-meta character and rest of characters in each Input pattern.**

**Program:**

%{

#include<stdio.h>

int n,m,t,c;

%}

%%

\n n++;

\t m++;

[ ]t++;

.c++;

%%

int yywrap(){return 1;}

int main(){

yylex();

printf("Total number of \nlines=%d \ntabs=%d \nspaces=%d \nchars=%d \n",n,m,t,c);

return 0;

}

**Output:**

Text

Description automatically generated

**PROGRAM-2**

**Design a LEX Code to identify and print valid Identifier of C/C++ in given Input pattern.**

**Program**

%{

#include<stdio.h>

int c=0;

%}

%%

^[a-z A-Z \_][a-z A-Z 0-9 \_]\* {c++;} printf("Valid Identifier");

.;

%%

int yywrap() {return 1;}

int main(){

yylex();

printf("Total valid identifiers=%d",c);

return 0;

}

**Output:**

Text

Description automatically generated

**PROGRAM-3**

**Design a LEX Code to identify and print integer and float value in given Input pattern.**

**Program:**

%{

#include<stdio.h>

int m=0,n=0;

%}

%%

[0-9]+ {m++; printf("\t Integer = %s",yytext);}

[0-9]\*"."[0-9]+ {n++; printf("\t Float = %s",yytext);}

. ;

%%

int yywrap(){return 1;}

int main(){

yylex();

printf("\nTotal number of Integer = %d & Float = %d \n",m,n);

return 0;

}

**Output:**

Text

Description automatically generated

**PROGRAM-4**

**Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, IDENTIFERS)**

**Program:**

%{

#include<stdio.h>

int n=0;

%}

%%

"while"|"if"|"else" {n++; printf("\t keywords: %s",yytext);}

"int"|"float" {n++; printf("\t keywords: %s",yytext);}

[a-zA-Z\_][a-zA-Z0-9\_]\* {n++; printf("\t Identifier: %s",yytext);}

"<="|"=="|"="|"++"|"-"|"\*"|"+" {n++; printf("\t operator: %s",yytext);}

"("|")"|"{"|"}"|","|";" {n++; printf("\t Seperators: %s",yytext);}

[0-9]\*"."[0-9]+ {n++; printf("\t Float %s",yytext);}

[0-9]+ {n++; printf("\t Integer: %s",yytext);}

. ;

%%

int yywrap(){return 1;}

int main()

{

yylex();

printf("\nTotal number of token = %d \n",n);

return 0;

}

**Output:**

Text

Description automatically generated

**PROGRAM-5**

**Design a LEX Code to count and print the number of total OPERATORS, SEPERATORS, KEYWORDS, IDENTIFERS in given ‘Input.txt’ file.**

**Program:**

%{

#include<stdio.h>

int Kc=0,Ic=0,Sc=0,Oc=0,Inc=0;

%}

%%

"while"|"if"|"else" {Kc++;}

"int"|"float" {Kc++;}

[a-zA-Z\_][a-zA-Z0-9\_]\* {Ic++;}

"<="|"=="|"="|"++"|"-"|"\*"|"+" {Oc++;}

[(){}|, ;] {Sc++;}

[0-9]\*"."[0-9]+ {Inc++;}

[0-9]+ {Inc++;}

%%

int yywrap(){return 1;}

int main() {

yylex();

extern FILE \*yyin,\*yyout;

yyin=fopen("input.txt","r");

yyout=fopen("output.txt","w");

{fprintf(yyout,"\n total no. of Keywords = %d\n",Kc);}

{fprintf(yyout,"\n total no. of Identifiers = %d\n",Ic);}

{fprintf(yyout,"\n total no. of Operators = %d\n",Oc);}

{fprintf(yyout,"\n total no. of Seperators = %d\n",Sc);}

{fprintf(yyout,"\n total no. of Integer = %d\n",Inc);}

return 0;

}

**Output:**

A screenshot of a video game

Description automatically generated with medium confidence

**PROGRAM-6**

**Design a LEX Code to count and print the number of total characters, words, white spaces in given ‘Input.txt’ file.**

**Program:**

%{

#include<stdio.h>

int Wc=0,WSc=0,Cc=0;

%}

%%

([a-zA-Z0-9]) {Wc++;}

[ ] {WSc++;}

. Cc++;

%%

int yywrap(){return 1;}

int main()

{

extern FILE \*yyin,\*yyout;

yyin=fopen("input.txt","r");

yyout=fopen("output.txt","w");

yylex();

{fprintf(yyout,"\n total no. of Words = %d\n",Wc);}

{fprintf(yyout,"\n total no. of White Spaces = %d\n",WSc);}

{fprintf(yyout,"\n total no. of Characters = %d\n",Cc);}

return 0;

}

**INPUT FILE:**

hello how are you

Geu

**OUTPUT FILE:**

total no. of Words = 5

total no. of White Spaces = 16

total no. of Characters = 36

**Output:**

A screenshot of a video game

Description automatically generated

**PROGRAM-7**

**Design a LEX Code to replace white spaces of ‘Input.txt’ file by a single blank character into ‘Output.txt’ file.**

**Program:**

%{

#include<stdio.h>

%}

%%

[\t" "]+ fprintf(yyout," ");

.|\n fprintf(yyout,"%s",yytext);

%%

int yywrap()

{

return 1;

}

int main(void)

{

yyin=fopen("input.txt","r");

yyout=fopen("output.txt","w");

yylex();

return 0;

}

**Input File:**a b c egf d

**Output File:**a b c egf d

**Output:**

A screenshot of a video game

Description automatically generated

**PROGRAM-8**

**Design a LEX Code to identify and print valid given string Identifier of ‘Input.txt’ file into ‘Output.txt’ file.**

**Program:**

%{

#include<stdio.h>

%}

%%

"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" fprintf(yyout,"this is a keyword");

^[a-z A-Z \_][a-z A-z 0-9 \_]\* fprintf(yyout,"Valid Identifier");

^[^a-z A-Z \_] fprintf(yyout,"not a valid identifier");

. ;

%%

int yywrap() {return 1;}

int main()

{

extern FILE \*yyin,\*yyout;

yyin=fopen("input.txt","r");

yyout=fopen("output.txt","w");

yylex();

return 0;

}

**Input FILE 1:**

Hellogeu

**Output FILE 1:**

Valid Identifier

**Input FILE 2:**

Hello

**Output FILE 2:**

Invalid Identifier

Invalid Identifier

Invalid Identifier

Invalid Identifier

Invalid Identifier

Output:

A screenshot of a computer

Description automatically generated with medium confidence

**PROGRAM-9**

**Design a LEX Code to remove the comments from any C-Program given at run-time and store into ‘out.txt’ file.**

**Program:**

%{

#include<stdio.h>

%}

%%

\/\/(.\*) {};

\/\\*(.\*\n)\*.\*\\*\/ {};

%%

int yywrap()

{

return 1;

}

int main()

{

yyin=fopen("input.txt","r");

yyout=fopen("output.txt","w");

yylex();

return 0;

}

**Input File:**

/\*Hey this is comment Section

We are going to remove this\*/

// this is single line comment

this is not a comment

**Output File:**

this is not a comment

**Output:**

A screenshot of a video game

Description automatically generated with medium confidence

**PROGRAM-10**

**Design a DFA in LEX Code which accepts string ending with 1 and not accept String ending with 0.**

**Program:**

%{

#include<stdio.h>

%}

%S A

%%

<INITIAL>0 BEGIN INITIAL;

<INITIAL>1 BEGIN A;

<INITIAL>\n {printf("Not Accepted");}

<A>0 BEGIN INITIAL;

<A>1 BEGIN A;

<A>\n {printf("Accepted");}

%%

int yywrap()

{

return 1;

}

int main()

{

yyin=fopen("input.txt","r");

yyout=fopen("output.txt","w");

yylex();

return 0;

}

**Input File 1:**

01101

**Output File 1:**

Accepted

**Input File 2:**

1001010

**Output File 2:**

Not Accepted

Output:

Graphical user interface, text

Description automatically generated