```
%x STATE
%x SINGLE_COMMENT
%x MULTI_COMMENT
%{
#include<iostream>
#include<stdlib.h>
#include<string.h>
#include "SymbolTable.h"
#include "y.tab.h"


using namespace std;

extern YYSTYPE yylval;
extern SymbolTable *table1;
extern SymbolTable *table2;
extern int scope;
extern FILE *out;
void yyerror(const char *);

int line_count=1;
int error_count=0;
char s1[100]="";


%}



digit [0-9]
letter [A-Za-z]
AlphaNumeric [a-zA-Z0-9]
delim [ \t\r\f\v]
newline \n
ws [delim]+
id (_|{letter})(_|{letter}|{digit})*
integer {digit}+
number {digit}*(\.{digit}+)?(E[+-]?{digit}+)?
ADDOP [+-]
MULOP [*/%]
UNDERSCORE _

%%

{delim}+    {}
{newline} {line_count++;}

"if"        {return IF;}
"else"      {return ELSE;}
"for"       {return FOR;}
"while"     {return WHILE;}
"int"       {return INT;}
"float"     {return FLOAT;}
"double"    {return DOUBLE;}
```

```
"char"          {return CHAR;}
"return"    {return RETURN;}
"void"          {return VOID;}
"main"          {return MAIN;}
"continue"  {return CONTINUE;}
"println"   {return PRINTLN;}

{ADDOP} {
                SymbolInfo *s= new  SymbolInfo(yytext, (char *)"ADDOP");
                yylval = (YYSTYPE)s;
                return ADDOP;
            }



{MULOP}     {
                SymbolInfo *s= new  SymbolInfo(yytext, (char *)"MULOP");
                yylval = (YYSTYPE)s;
                return MULOP;
            }


"="   {return ASSIGNOP;}

">"  |
"<"  |
">=" |
"<=" |
"==" |
"!=" {
            SymbolInfo *s= new  SymbolInfo(yytext, (char *)"RELOP");
            yylval = (YYSTYPE)s;
            return RELOP;
      }

"&&"  |
"||"  {
                SymbolInfo *s= new  SymbolInfo(yytext, (char
*)"LOGICOP");
                yylval = (YYSTYPE)s;
                return LOGICOP;
            }
"!"         {     return NOT; }


";"   {return SEMICOLON;}
","   {return COMMA;}
"("   {return LPAREN;}
")"   {return RPAREN;}
"{"   {return LCURL;}
"}"   {return RCURL;}
"["   {return LTHIRD;}
"]"   {return RTHIRD;}
"++"  {return INCOP;}
```

```
"--"  {return DECOP;}



{integer}   {
                        SymbolInfo *s= new  SymbolInfo(yytext, (char
*)"CONST_INT");
                        s->dataType = 0;
                        s->v.i = atoi(yytext);
                        yylval = (YYSTYPE)s;
                        return CONST_INT;
                  }
{number}    {
                        SymbolInfo *s= new  SymbolInfo(yytext, (char
*)"CONST_FLOAT");
                        s->dataType = 1;
                        s->v.i = atoi(yytext);
                        yylval = (YYSTYPE)s;
                        return CONST_FLOAT;
                  }
{id}        {

                        SymbolInfo *s;
                        if(scope==1) {s=table1->Lookup(yytext);}
                        else s=table2->Lookup(yytext);
                        if(s==NULL)
                        {
                              s= new SymbolInfo(yytext, (char *)"ID");
                        }
                        yylval = (YYSTYPE)s;

                        return ID;
                  }
({digit}+)({letter}|{UNDERSCORE})+      {
                        error_count++;
                        fprintf(out,"\nError at line %d: Invalid prefix on
ID or invalid suffix on Number %s\n",line_count,yytext);
                  }
\'{AlphaNumeric}{AlphaNumeric}+\' {
                        error_count++;
                        fprintf(out,"\nError at line %d: Ill formed
character %s\n",line_count,yytext);
                  }
''                {
                        error_count++;
                        fprintf(out,"\nError at line no.%d Empty character
constant error %s\n",line_count,yytext);
}
\'{AlphaNumeric}*$      {
                        error_count++;
                        fprintf(out,"\nError at line %d: Unterminated
character %s\n",line_count,yytext);
                  }
```

```
{digit}*(\.{digit}*)?(E[+-]?{digit}*)((\.{digit}*)(E[+-]?{digit}+)?)*
      {
                        error_count++;
                        fprintf(out,"\nError at line %d: Ill formed number
%s\n",line_count,yytext);
                  }

{digit}*(\.{digit}*)(\.{digit}*)+ {
                        error_count++;
                        fprintf(out,"\nError at line %d: Too many decimal
point %s\n",line_count,yytext);
                  }

'(.)+'                  {
                        error_count++;
                        fprintf(out,"Error at line:%d: Multi character
constant error %s\n",line_count,yytext);
                  }

'[^(\')|(\n)]*          {
                        error_count++;
                        fprintf(out,"Error at line:%d: Unterminated
character %s\n",line_count,yytext);
                  }

\"    {BEGIN STATE;
      strcat(s1,yytext);}
<STATE>{newline} {error_count++;
                  fprintf(out,"Error at line no. %d Unterminated String
%s\n",line_count,s1);
                  strcpy(s1,"");
                  line_count++;
                  BEGIN INITIAL;}
<STATE>\\t  {  strcat(s1,"\\t");}
<STATE>\\n  {  strcat(s1,"\\n");}
<STATE>\\a  {  strcat(s1,"\\n");}
<STATE>\\b  {  strcat(s1,"\\b");}
<STATE>\\f  {  strcat(s1,"\\f");}
<STATE>\\v  {  strcat(s1,"\\v");}
<STATE>\\r  {  strcat(s1,"\\r");}
<STATE>\\   {  strcat(s1,yytext);}

<STATE>\\{newline}      {line_count++;strcat(s1,yytext);}


<STATE>\"   {strcat(s1,yytext);
            fprintf(out,"Line no. %d: Token <STRING> Lexeme %s
found\n",line_count,s1);
            strcpy(s1,"");
            BEGIN INITIAL;
}
<STATE>[^\\\"]    {  strcat(s1,yytext);}

\/\/  { BEGIN SINGLE_COMMENT;
```

```
      strcat(s1,yytext);}
<SINGLE_COMMENT>{newline}    {
                        fprintf(out,"Line no. %d: Token <SINGLE COMMENT>
Lexeme %s found\n",line_count,s1);
                        strcpy(s1,"");
                        BEGIN INITIAL;
                        }
<SINGLE_COMMENT>(\\)+  {strcat(s1,yytext);}
<SINGLE_COMMENT>(\\)+{newline}    {line_count++; strcat(s1,yytext);}

<SINGLE_COMMENT>[^\\]  {strcat(s1,yytext);}

\/\*  { BEGIN MULTI_COMMENT;
      strcat(s1,yytext);}
<MULTI_COMMENT>(\*)+   {strcat(s1,yytext);}
<MULTI_COMMENT>(\*)+\/ {     strcat(s1,yytext);
                        fprintf(out,"Line no. %d: Token <MULTI COMMENT>
Lexeme %s found\n",line_count,s1);
                        line_count++;
                        strcpy(s1,"");
                        BEGIN INITIAL;
                  }
<MULTI_COMMENT>{newline}     {line_count++; strcat(s1,"\n");}

<MULTI_COMMENT>[^\*]         {strcat(s1,yytext);}

<MULTI_COMMENT><<EOF>> {     error_count++;
                        fprintf(out,"Error at line no: %d: Unterminated
comment  %s \n",line_count,s1);
                        strcpy(s1,"");
                        BEGIN INITIAL;
                        }

%%



int yywrap(void)
{
      return 1;
}
```