**CS-224 Formal Languages & Theory of Automata**

**Semester Project Report**

**Ghulam Ishaq Khan Institute of Engineering Sciences & Technology, Topi, Swabi 23460, Pakistan**

**ARH Compiler**

**Members:**

* 2021051 (Adeen Amir)
* 2021438 (Rehan Riaz)
* 2021193 (Hamza Zuberi)

**Lexical Analyzer Code:**

%option *noyywrap*

%{

    #include <stdio.h>

    #include <stdlib.h>

    #include <string.h>

    int lineno = 1;

    void printToken(char \**TOKEN*);

    FILE \* oFile;

    FILE \*iFile;

    void printOut(FILE \**oFile*);

    void yyerror();

%}

%x ML\_COMMENT

alpha       [a-zA-Z]

digit       [0-9]

alnum       {alpha}|{digit}

print       [ -~]

VARIABLE        {alpha}+{alnum}\*

INTCONST        "0"|[0-9]{digit}\*

END\_LINE        (\'\\[nftrbv]\')

STRING      \"{print}\*\"

PREPROCESSOR "#include <"+{alpha}\*+">"|"#include \""+{alnum}\*+"."{alpha}\*"\""|"#include <"+{alnum}\*+"."{alpha}\*""

DEFINE      "#define "+{alnum}\*

%%

"//".\*                  { printf("comment out single line %d\n", lineno); }

"/\*"                    { printf("Comment out starting line %d ", lineno); BEGIN(ML\_COMMENT); }

<ML\_COMMENT>"\*/"        { printf("to line %d\n", lineno); BEGIN(INITIAL); }

<ML\_COMMENT>[^\*\n]+

<ML\_COMMENT>"\*"

<ML\_COMMENT>"\n"        { lineno += 1; }

"using namespace std"   {printToken("using\_namespace\_std");}

"switch"|"SWITCH"       { printToken("KW\_SWITCH"); }

"case"|"CASE"           { printToken("KW\_CASE"); }

"bool"|"BOOL"           { printToken("KW\_BOOL"); }

"operator"|"OPERATOR"   { printToken("KW\_OPERATOR"); }

"cout"|"COUT"           { printToken("KW\_COUT"); }

"cin"|"CIN"             { printToken("KW\_CIN"); }

"char"|"CHAR"           { printToken("KW\_CHAR"); }

"friend"|"FRIEND"       { printToken("KW\_FRIEND"); }

"delete"|"DELETE"       { printToken("KW\_DELETE"); }

"long"|"LONG"           { printToken("KW\_LONG"); }

"static"|"STATIC"       { printToken("KW\_STATIC"); }

"const"|"CONST"         { printToken("KW\_CONST"); }

"string"|"STRING"       { printToken("KW\_STRING"); }

"int"|"INT"             { printToken("KW\_INT"); }

"float"|"FLOAT"         { printToken("KW\_FLOAT"); }

"double"|"DOUBLE"       { printToken("KW\_DOUBLE"); }

"if"|"IF"               { printToken("KW\_IF"); }

"else"|"ELSE"           { printToken("KW\_ELSE"); }

"while"|"WHILE"         { printToken("KW\_WHILE"); }

"do"|"DO"               { printToken("KW\_DO"); }

"for"|"FOR"             { printToken("KW\_FOR"); }

"continue"|"CONTINUE"   { printToken("KW\_CONTINUE"); }

"break"|"BREAK"         { printToken("KW\_BREAK"); }

"void"|"VOID"           { printToken("KW\_VOID"); }

"return"|"RETURN"       { printToken("KW\_RETURN"); }

"public"|"PUBLIC"       { printToken("KW\_PUBLIC"); }

"private"|"PRIVATE"     { printToken("KW\_PRIVATE"); }

"protected"|"PROTECTED" { printToken("KW\_PROTECTED"); }

"+"                     { printToken("ADDOP"); }

"-"                     { printToken("SUBOP"); }

"\*"                     { printToken("MULOP"); }

"/"                     { printToken("DIVOP"); }

"++"                    { printToken("INCR"); }

"--"                    { printToken("INCR"); }

"||"                    { printToken("OROR"); }

"|"                     { printToken("OR"); }

"&&"                    { printToken("ANDOP"); }

"!"                     { printToken("NOTOP"); }

"=="|"!="               { printToken("EQUOP"); }

">"|"<"|">="|"<="       { printToken("RELOP"); }

"<<"                    { printToken("INSERTIONOP"); }

">>"                    { printToken("EXTRACTIONOP"); }

"->"                    { printToken("ARROW"); }

"("             { printToken("LPAREN"); }

")"             { printToken("RPAREN"); }

"'"             { printToken("SINGLEQUOTATION"); }

"]"             { printToken("LBRACK"); }

"\\"            { printToken("BACKSLASH"); }

"["             { printToken("RBRACK"); }

"~"             { printToken("TILDE"); }

"{"             { printToken("LBRACE"); }

"}"             { printToken("RBRACE"); }

"\_"             { printToken("UNDERSCORE"); }

";"             { printToken("SEMI"); }

"."             { printToken("DOT"); }

":"             { printToken("COLON"); }

","             { printToken("COMMA"); }

"="             { printToken("ASSIGN"); }

"&"             { printToken("REFER"); }

{VARIABLE}      { printToken("VARIABLE"); }

{INTCONST}      { printToken("INT\_CONST"); }

{END\_LINE}      { printToken("END\_LINE"); }

{STRING}        { printToken("STRING"); }

{PREPROCESSOR}  { printToken("PREPROCESSOR");}

{DEFINE}        { printToken("DEFINE");}

"\n"            { lineno += 1; }

[ \t\r\f]+

.               { yyerror("Unrecognized character"); }

%%

void printToken(char \**TOKEN*){

    printf("Token: %s\ttoken\_id: %s\tlineno: %d\n", yytext, TOKEN, lineno);

    fprintf(oFile,"%s, %s, %d\n", yytext, TOKEN, lineno);

}

void yyerror(char \**message*){

    printf("Error: \"%s\" in line %d. Token = %s\n", message, lineno, yytext);

    exit(1);

}

int main(int *argc*, char \**argv*[]){

    yyin = fopen("case4.cpp","r");

    if(yyin==NULL)

    {

        printf("ERROR Input source not found\n");

        exit(1);

    }

    oFile=fopen("output.csv", "w");

    yylex();

    fclose(yyin);

    return 0;

}

**Sample Example 1:**

# #include <iostream>

using namespace std;

int main()

{

    int n, b;

    cin >> n >> b;

    cout << "Multiplied Value" << n \* b;

}

**Output:**

|  |  |  |
| --- | --- | --- |
| **Token** | **Token ID** | **Line No.** |
| #include <iostream> | PREPROCESSOR | 1 |
| using namespace std | using\_namespace\_std | 3 |
| ; | SEMI | 3 |
| int | KW\_INT | 5 |
| main | VARIABLE | 5 |
| ( | LPAREN | 5 |
| ) | RPAREN | 5 |
| { | LBRACE | 6 |
| int | KW\_INT | 7 |
| n | VARIABLE | 7 |
| , | COMMA | 7 |
| b | VARIABLE | 7 |
| ; | SEMI | 7 |
| cin | KW\_CIN | 8 |
| >> | EXTRACTIONOP | 8 |
| n | VARIABLE | 8 |
| >> | EXTRACTIONOP | 8 |
| b | VARIABLE | 8 |
| ; | SEMI | 8 |
| cout | KW\_COUT | 9 |
| << | INSERTIONOP | 9 |
| Multiplied Value | STRING | 9 |
| << | INSERTIONOP | 9 |
| n | VARIABLE | 9 |
| \* | MULOP | 9 |
| b | VARIABLE | 9 |
| ; | SEMI | 9 |
| } | RBRACE | 10 |

**Sample Example 2:**

#include <iostream>

using namespace std;

int main()

{

    int n = 5;

    cin >> n;

    if (n > 10)

    {

        if (n > 15)

            cout << "YES" << endl;

        else

            cout << "NO" << endl;

    }

    else

        cout << "hello";

}

**Output:**

|  |  |  |
| --- | --- | --- |
| **Token** | **Token ID** | **Line No.** |
| #include <iostream> | PREPROCESSOR | 1 |
| using namespace std | using\_namespace\_std | 3 |
| ; | SEMI | 3 |
| int | KW\_INT | 5 |
| main | VARIABLE | 5 |
| ( | LPAREN | 5 |
| ) | RPAREN | 5 |
| { | LBRACE | 6 |
| int | KW\_INT | 7 |
| n | VARIABLE | 7 |
| = | ASSIGN | 7 |
| 5 | INT\_CONST | 7 |
| ; | SEMI | 7 |
| cin | KW\_CIN | 8 |
| >> | EXTRACTIONOP | 8 |
| n | VARIABLE | 8 |
| ; | SEMI | 8 |
| if | KW\_IF | 9 |
| ( | LPAREN | 9 |
| n | VARIABLE | 9 |
| > | RELOP | 9 |
| 10 | INT\_CONST | 9 |
| ) | RPAREN | 9 |
| { | LBRACE | 10 |
| if | KW\_IF | 11 |
| ( | LPAREN | 11 |
| n | VARIABLE | 11 |
| > | RELOP | 11 |
| 15 | INT\_CONST | 11 |
| ) | RPAREN | 11 |
| { | LBRACE | 12 |
| cout | KW\_COUT | 13 |
| << | INSERTIONOP | 13 |
| YES | STRING | 13 |
| << | INSERTIONOP | 13 |
| endl | VARIABLE | 13 |
| ; | SEMI | 13 |
| } | RBRACE | 14 |
| else | KW\_ELSE | 15 |
| { | LBRACE | 16 |
| cout | KW\_COUT | 17 |
| << | INSERTIONOP | 17 |
| NO | STRING | 17 |
| << | INSERTIONOP | 17 |
| endl | VARIABLE | 17 |
| ; | SEMI | 17 |
| } | RBRACE | 18 |
| } | RBRACE | 19 |
| else | KW\_ELSE | 20 |
| { | LBRACE | 21 |
| cout | KW\_COUT | 22 |
| << | INSERTIONOP | 22 |
| hello | STRING | 22 |
| ; | SEMI | 22 |
| } | RBRACE | 23 |
| } | RBRACE | 24 |

**Sample Example 3:**

#include <iostream>

using namespace std;

#define SIZE 5

int main()

{

    int arr[SIZE][SIZE];

    for (int i = 0; i < SIZE; i++)

        for (int j = 0; j < SIZE; j++)

            cin >> arr[i][j];

    for (int i = 0; i < SIZE; i++)

    {

        for (int j = 0; j < SIZE; j++)

            cout << arr[i][j]<<" ";

        cout << "\n";

    }

}

**Output:**

|  |  |  |
| --- | --- | --- |
| **Token** | **Token ID** | **Line No.** |
| #include <iostream> | PREPROCESSOR | 1 |
| using namespace std | using\_namespace\_std | 2 |
| ; | SEMI | 2 |
| #define SIZE | DEFINE | 3 |
| 5 | INT\_CONST | 3 |
| int | KW\_INT | 5 |
| main | VARIABLE | 5 |
| ( | LPAREN | 5 |
| ) | RPAREN | 5 |
| { | LBRACE | 6 |
| int | KW\_INT | 7 |
| arr | VARIABLE | 7 |
| [ | RBRACK | 7 |
| SIZE | VARIABLE | 7 |
| ] | LBRACK | 7 |
| [ | RBRACK | 7 |
| SIZE | VARIABLE | 7 |
| ] | LBRACK | 7 |
| ; | SEMI | 7 |
| for | KW\_FOR | 8 |
| ( | LPAREN | 8 |
| int | KW\_INT | 8 |
| i | VARIABLE | 8 |
| = | ASSIGN | 8 |
| 0 | INT\_CONST | 8 |
| ; | SEMI | 8 |
| i | VARIABLE | 8 |
| < | RELOP | 8 |
| SIZE | VARIABLE | 8 |
| ; | SEMI | 8 |
| i | VARIABLE | 8 |
| ++ | INCR | 8 |
| ) | RPAREN | 8 |
| for | KW\_FOR | 9 |
| ( | LPAREN | 9 |
| int | KW\_INT | 9 |
| j | VARIABLE | 9 |
| = | ASSIGN | 9 |
| 0 | INT\_CONST | 9 |
| ; | SEMI | 9 |
| j | VARIABLE | 9 |
| < | RELOP | 9 |
| SIZE | VARIABLE | 9 |
| ; | SEMI | 9 |
| j | VARIABLE | 9 |
| ++ | INCR | 9 |
| ) | RPAREN | 9 |
| cin | KW\_CIN | 10 |
| >> | EXTRACTIONOP | 10 |
| arr | VARIABLE | 10 |
| [ | RBRACK | 10 |
| i | VARIABLE | 10 |
| ] | LBRACK | 10 |
| [ | RBRACK | 10 |
| j | VARIABLE | 10 |
| ] | LBRACK | 10 |
| ; | SEMI | 10 |
| for | KW\_FOR | 12 |
| ( | LPAREN | 12 |
| int | KW\_INT | 12 |
| i | VARIABLE | 12 |
| = | ASSIGN | 12 |
| 0 | INT\_CONST | 12 |
| ; | SEMI | 12 |
| i | VARIABLE | 12 |
| < | RELOP | 12 |
| SIZE | VARIABLE | 12 |
| ; | SEMI | 12 |
| i | VARIABLE | 12 |
| ++ | INCR | 12 |
| ) | RPAREN | 12 |
| { | LBRACE | 13 |
| for | KW\_FOR | 14 |
| ( | LPAREN | 14 |
| int | KW\_INT | 14 |
| j | VARIABLE | 14 |
| = | ASSIGN | 14 |
| 0 | INT\_CONST | 14 |
| ; | SEMI | 14 |
| j | VARIABLE | 14 |
| < | RELOP | 14 |
| SIZE | VARIABLE | 14 |
| ; | SEMI | 14 |
| j | VARIABLE | 14 |
| ++ | INCR | 14 |
| ) | RPAREN | 14 |
| cout | KW\_COUT | 15 |
| << | INSERTIONOP | 15 |
| arr | VARIABLE | 15 |
| [ | RBRACK | 15 |
| i | VARIABLE | 15 |
| ] | LBRACK | 15 |
| [ | RBRACK | 15 |
| j | VARIABLE | 15 |
| ] | LBRACK | 15 |
| << | INSERTIONOP | 15 |
|  | STRING | 15 |
| ; | SEMI | 15 |
| cout | KW\_COUT | 17 |
| << | INSERTIONOP | 17 |
| \n | STRING | 17 |
| ; | SEMI | 17 |
| } | RBRACE | 18 |
| } | RBRACE | 19 |

**Sample Example 4:**

#include <iostream>

using namespace std;

class *CMyString*

{

private:

    int size;

    char \*String;

public:

    CMyString(*string* *a*)

    {

        size = *a*.size();

        String = **new** char[size + 1];

    }

    CMyString()

    {

        size = 50;

        String = **new** char[size];

    }

};

int main()

{

*CMyString* a("ABCDEF");

*CMyString* b("1234567");

  return 0;

}

**Output:**

|  |  |  |
| --- | --- | --- |
| **Token** | **Token ID** | **Line No.** |
| #include <iostream> | PREPROCESSOR | 1 |
| using namespace std | using\_namespace\_std | 2 |
| ; | SEMI | 2 |
| class | VARIABLE | 3 |
| CMyString | VARIABLE | 3 |
| { | LBRACE | 4 |
| private | KW\_PRIVATE | 5 |
| : | COLON | 5 |
| int | KW\_INT | 6 |
| size | VARIABLE | 6 |
| ; | SEMI | 6 |
| char | KW\_CHAR | 7 |
| \* | MULOP | 7 |
| String | VARIABLE | 7 |
| ; | SEMI | 7 |
| public | KW\_PUBLIC | 9 |
| : | COLON | 9 |
| CMyString | VARIABLE | 10 |
| ( | LPAREN | 10 |
| string | KW\_STRING | 10 |
| a | VARIABLE | 10 |
| ) | RPAREN | 10 |
| { | LBRACE | 11 |
| size | VARIABLE | 12 |
| = | ASSIGN | 12 |
| a | VARIABLE | 12 |
| . | DOT | 12 |
| size | VARIABLE | 12 |
| ( | LPAREN | 12 |
| ) | RPAREN | 12 |
| ; | SEMI | 12 |
| String | VARIABLE | 13 |
| = | ASSIGN | 13 |
| new | VARIABLE | 13 |
| char | KW\_CHAR | 13 |
| [ | RBRACK | 13 |
| size | VARIABLE | 13 |
| + | ADDOP | 13 |
| 1 | INT\_CONST | 13 |
| ] | LBRACK | 13 |
| ; | SEMI | 13 |
| } | RBRACE | 14 |
| CMyString | VARIABLE | 16 |
| ( | LPAREN | 16 |
| ) | RPAREN | 16 |
| { | LBRACE | 17 |
| size | VARIABLE | 18 |
| = | ASSIGN | 18 |
| 50 | INT\_CONST | 18 |
| ; | SEMI | 18 |
| String | VARIABLE | 19 |
| = | ASSIGN | 19 |
| new | VARIABLE | 19 |
| char | KW\_CHAR | 19 |
| [ | RBRACK | 19 |
| size | VARIABLE | 19 |
| ] | LBRACK | 19 |
| ; | SEMI | 19 |
| } | RBRACE | 20 |
| } | RBRACE | 22 |
| ; | SEMI | 22 |
| int | KW\_INT | 24 |
| main | VARIABLE | 24 |
| ( | LPAREN | 24 |
| ) | RPAREN | 24 |
| { | LBRACE | 25 |
| CMyString | VARIABLE | 26 |
| a | VARIABLE | 26 |
| ( | LPAREN | 26 |
| ABCDEF | STRING | 26 |
| ) | RPAREN | 26 |
| ; | SEMI | 26 |
| CMyString | VARIABLE | 27 |
| b | VARIABLE | 27 |
| ( | LPAREN | 27 |
| 1234567 | STRING | 27 |
| ) | RPAREN | 27 |
| ; | SEMI | 27 |
| } | RBRACE | 29 |