CD MINI PROJECT

TOKEN GENERATOR
FINAL LEXICAL ANALYSIS

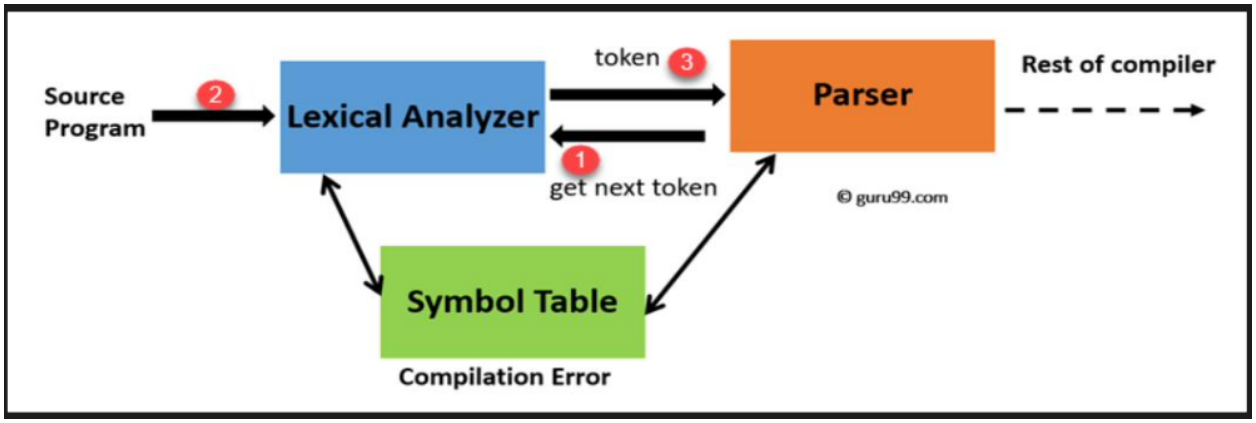
UTKARSH SINHA , MALLIKA SHARDA , ADITYA SINHA

INTRODUCTION

LEXICAL ANALYSIS is the very first phase in the compiler designing. It takes the modified source code which is written in the form of sentences. In other words, it helps you to convert a sequence of characters into a sequence of tokens. The lexical analyser breaks this syntax into a series of tokens. It removes any extra space or comment written in the source code.

Programs that perform lexical analysis are called lexical analysers. A lexical analyser contains tokenizer or scanner. If the lexical analyser detects that the token is invalid, it generates an error. It reads character streams from the source code, checks for legal tokens, and pass the data to the syntax analyser when it demands.

ARCHITECTURE AND WORKING



Code:

```
#include <bits/stdc++.h>
#include <regex>
#include<time.h>
#include <iterator>
#define deb(x) cout<<#x<<" = "<<x<<endl

using namespace std;

map<string,string> Make_Regex_Map(){
   map<string,string> my_map {
      { "\\;|\\{|\\}|\\(|\\)|\\,|\\#", "Special Symbol"},
      { "int|char|float|bool|cin|cout|main|using|namespace|std","Keywords"},
      { "\\include|define", "Pre-Processor Directive"},
      {"\\iostream|\\stdio|\\string","Library"},
      { "\\*|\\+|\\>>|\\<<|<|>",  "Operator"},
      { "[0-9]+" ,   "Integer" },
      { "[^include][^iostream][^int][^main][^cin][^cout][^;][^>>][^,][^[B ;cin]][a-z]+" ,
"Identifier" },
      { "[A-Z]+",    "Variable"},
      {"[ ]",""},
   };
   return my_map;
}

map<size_t,pair<string,string>> Match_Language (map<string,string> patterns,string
str){

 map< size_t, pair<string,string> > lang_matches;

   for ( auto i = patterns.begin(); i != patterns.end(); ++i )
   {
      regex compare(i->first);
      auto words_begin = sregex_iterator( str.begin(), str.end(), compare );
      auto words_end   = sregex_iterator();
   //MAKING PAIRS OF [STRING OF REGEX 'compare' : 'pattern']
      for ( auto it = words_begin; it != words_end; ++it )
         lang_matches[ it->position() ] = make_pair( it->str(), i->second );
   }
```

```
    return lang_matches;
}

string tell_Lexeme(string op){
    if(op=="*") return "MUL";
    else if(op=="+") return "ADD";
    else if(op==">>") return "INS";
    else if(op=="<<") return "EXTR";
    else if(op==">") return "RSHFT";
    else if(op=="<") return "LSHFT";
    else return "";
}
int main()
{
    ofstream fout;
    cout<<endl<<endl<<endl;
    cout.fill(' ');
    cout.width(100);
    fout.open("OutputFile");
    char c;
    string filename;


    cout<<"ENTER THE SOURCE CODE FILE NAME: Example \"abc.txt\" \n";
    cin>>filename;
    fstream fin(filename, fstream::in);
    string str;
    //Fetching Source Code in String type 'str'
    if(fin.is_open()){
    while(fin>> noskipws>>c)
        str=str+c;

    //Making a map which which will define the regex in source code to its pattern in
my language.
    map<string,string> patterns =Make_Regex_Map();

    /*DECLARING MAP 'lang_matches' from 'patterns' map which will pair up the
patterns
    from the ['Source Code':'Defined Pattern' via a Regex named 'compare'. */
    map< size_t, pair<string,string> > lang_matches = Match_Language(patterns,str);

    // Writing matches in File ignoring 'spaces' and '\n'.
```

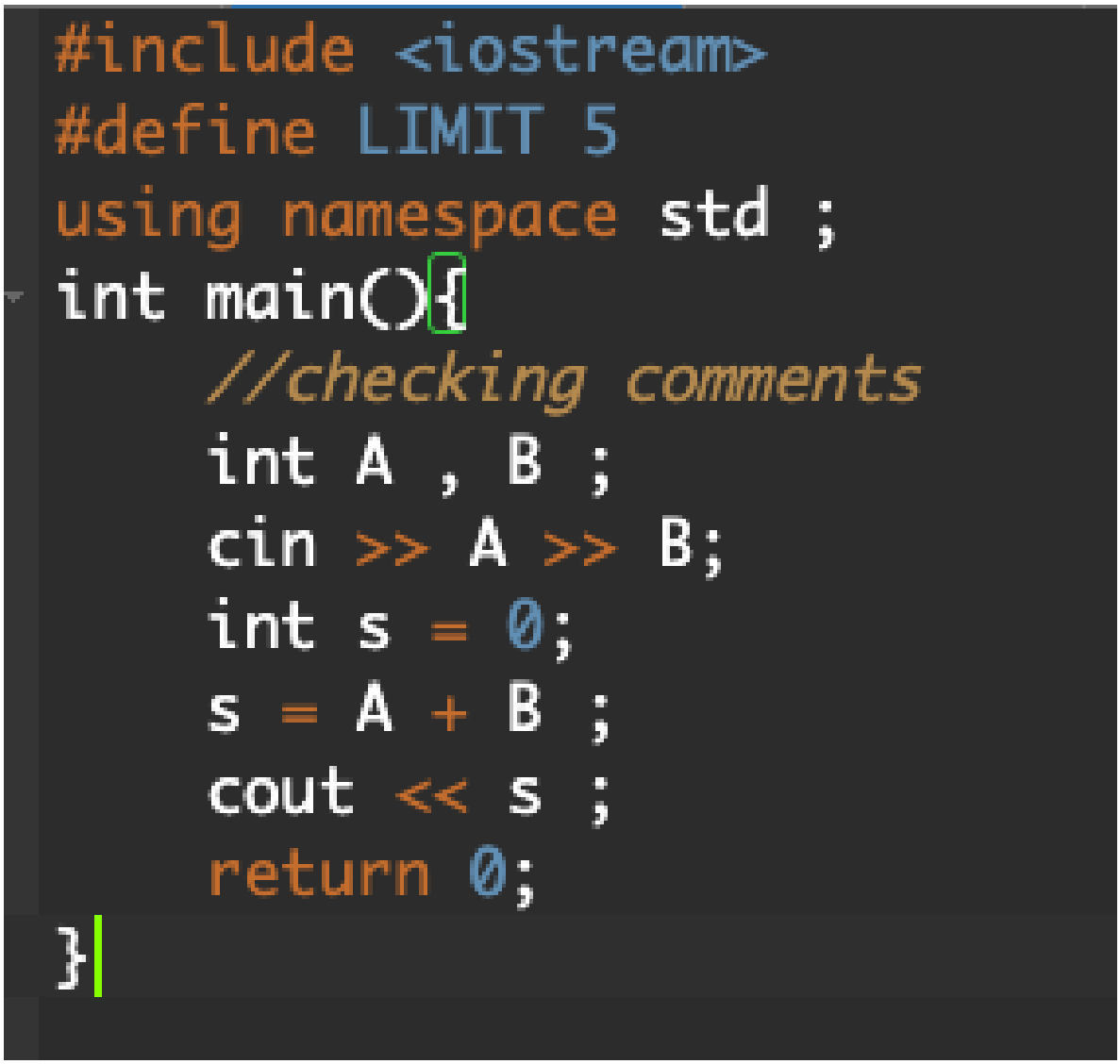
```
   int count = 1;

cout<<"\t\t\t\t----------------------------------------------------------------------------------------------------
---- \n";
   cout.width(40);
   cout<<"\t       NUMBER"<<setw(10)<<"            TOKEN "<< " "<< "           "<<setw(20)<<"
PATTERN \n";
   cout.fill(' ');
   cout.width(40);


cout<<"\t\t\t\t----------------------------------------------------------------------------------------------------
---- \n\n\n";
   //cout<<"\t\t\t\t                        PROCESSING SOURCE CODE.....
\n\n\n";
   //Sleep(5000);
   for ( auto match = lang_matches.begin(); match != lang_matches.end(); ++match ){

      if(!(match->second.first==" ")&&!(match->second.first=="//")){

         if(match->second.second=="Variable"||match->second.second=="Identifier"){

         cout.width(40);
         if(count<10){
            string double_digits = to_string(count);
            double_digits = "0"+double_digits;
            cout<<"\t Token   No :"<<double_digits<< "  |   "<<setw(10)<<
match->second.first << " " <<" ------->  |"<< setw(25)<< match->second.second
<<setw(18)<<" ,  POINTER TO SYMBOL TABLE    "<<endl;
            fout<<"\t Token   No :"<<double_digits<< "  |   "<<setw(10)<<
match->second.first << " " <<" ------->  |"<< setw(25)<< match->second.second
<<setw(18)<<" ,  POINTER TO SYMBOL TABLE    "<<endl;
         }
         else{
            cout<<"\t Token   No :"<<count<< "  |   "<< setw(10)<< match->second.first << " "
<<" ------->  |"<< setw(25) << match->second.second  <<setw(18)<<" ,  POINTER TO
SYMBOL TABLE    "<<endl;
            fout<<"\t Token   No :"<<count<< "  |   "<< setw(10)<< match->second.first << " "
<<" ------->  |"<< setw(25) << match->second.second  <<setw(18)<<" ,  POINTER TO
SYMBOL TABLE    "<<endl;
         }
         count++;
```

```
        }

        else{
           if(match->second.second=="Operator"){
           cout.width(40);
           string op=tell_Lexeme(match->second.first);
            if(count<10){
           string double_digits = to_string(count);
           double_digits = "0"+double_digits;
           cout<<"\t Token   No :"<<double_digits<< "  |   "<< setw(10)<<
match->second.first << " " <<" -------->  |"<< setw(25)<< match->second.second<<" ,
"<<op<<"    "  <<endl;
           fout<<"\t Token   No :"<<double_digits<< "  |   "<< setw(10)<<
match->second.first << " " <<" -------->  |"<< setw(25)<< match->second.second<<" ,
"<<op<<"    "  <<endl;
           count++;
           }
           else{
              cout<<"\t Token   No :"<<count<< "  |   "<< setw(10)<< match->second.first <<
" " <<" -------->  |"<< setw(25)<< match->second.second<<" , "<<op<<"    "  <<endl;
              fout<<"\t Token   No :"<<count<< "  |   "<< setw(10)<< match->second.first <<
" " <<" -------->  |"<< setw(25)<< match->second.second<<" , "<<op<<"    "  <<endl;
              count++;
           }

           }
           else{
              cout.width(40);
              if(count<10){
              string double_digits = to_string(count);
              double_digits = "0"+double_digits;
              cout<<"\t Token   No :"<<double_digits<< "  |   "<< setw(10)<<
match->second.first << " " <<" -------->  |"<< setw(25)<< match->second.second<<"    "
<<endl;
              fout<<"\t Token   No :"<<double_digits<< "  |   "<< setw(10)<<
match->second.first << " " <<" -------->  |"<< setw(25)<< match->second.second<<"    "
<<endl;
              count++;
              }
              else{
                  cout<<"\t Token   No :"<<count<< "  |   "<<setw(10)<< match->second.first
<< " " <<" -------->  |"<< setw(25)<< match->second.second<<"    "  <<endl;
```

```
                    fout<<"\t Token   No :"<<count<< "  |   "<<setw(10)<< match->second.first
<< " " <<" ------->  |"<< setw(25)<< match->second.second<<"    "  <<endl;
                    count++;
                }

            }

        }
    }
  }

  string command= " ";

  while(command !="EXIT"){
  cout.fill(' ');
  cout.width(40);
  cout<<"\n\n\t PRESS TYPE `EXIT` TO CLOSE WINDOW.\n\t NOTE: AN OUTPUT FILE
WILL BE GENERATED IN THE SAME FOLDER AS `Output.txt` \n";
  cin.width(40);
  cin>>command;

  if(command == "exit"||command == "EXIT"|| command == "Exit")
  break;

  else{
     cout.fill(' ');
     cout.width(40);
     cout<<"Please enter correct word.";
     cin.width(10);
     cin>>command;
  }

  }
  }

  else{
     cout.fill(' ');
  cout.width(40);
  cout<<"\n FILE NOT FOUND!\n\n";
  }
  return 0;
}
```

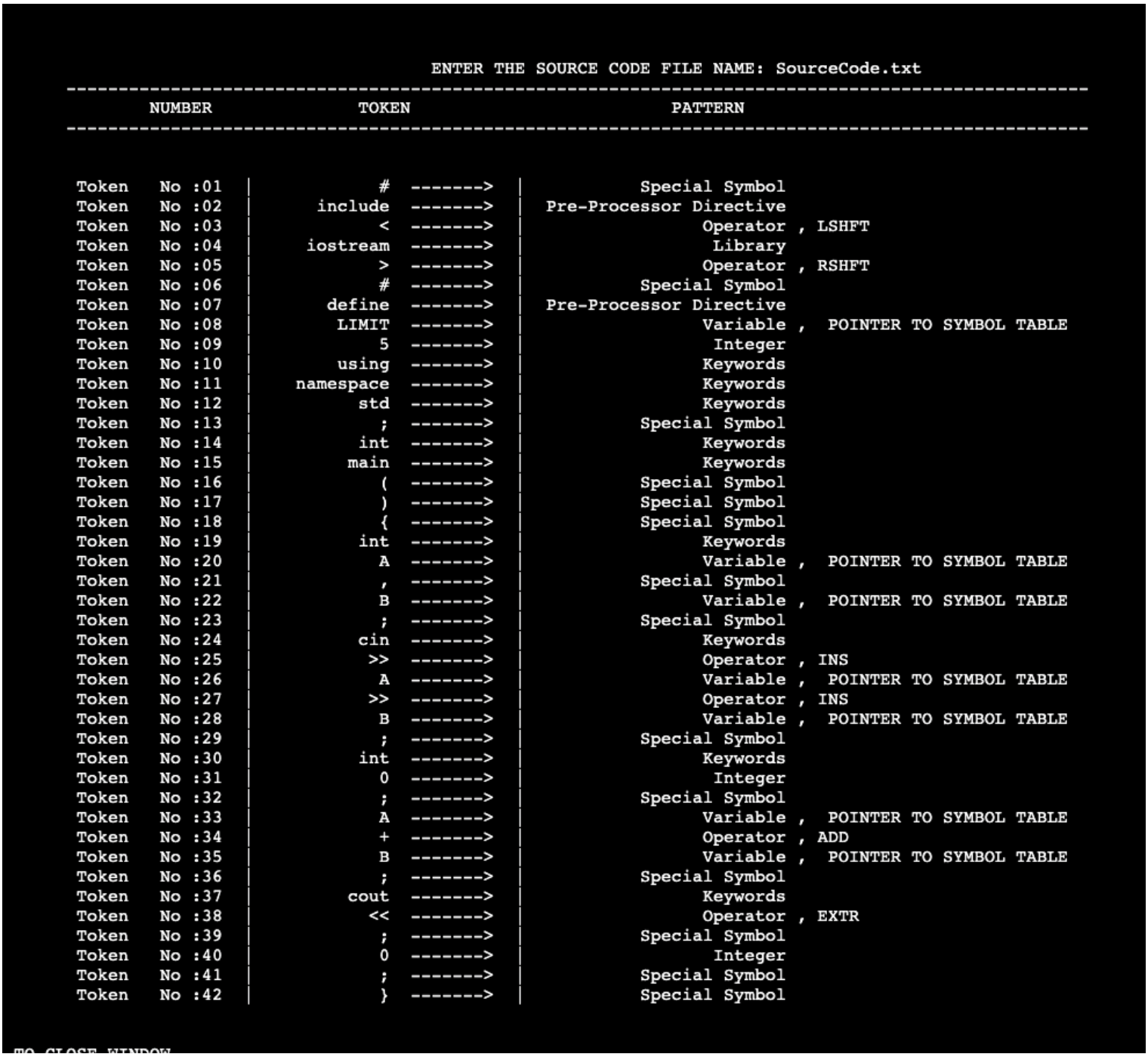
Source code (SourceCode.txt)

A program in C++ to take 2 integers as input and print the sum



```cpp
#include <iostream>
#define LIMIT 5
using namespace std ;
int main(){
    //checking comments
    int A , B ;
    cin >> A >> B;
    int s = 0;
    s = A + B ;
    cout << s ;
    return 0;
}
```

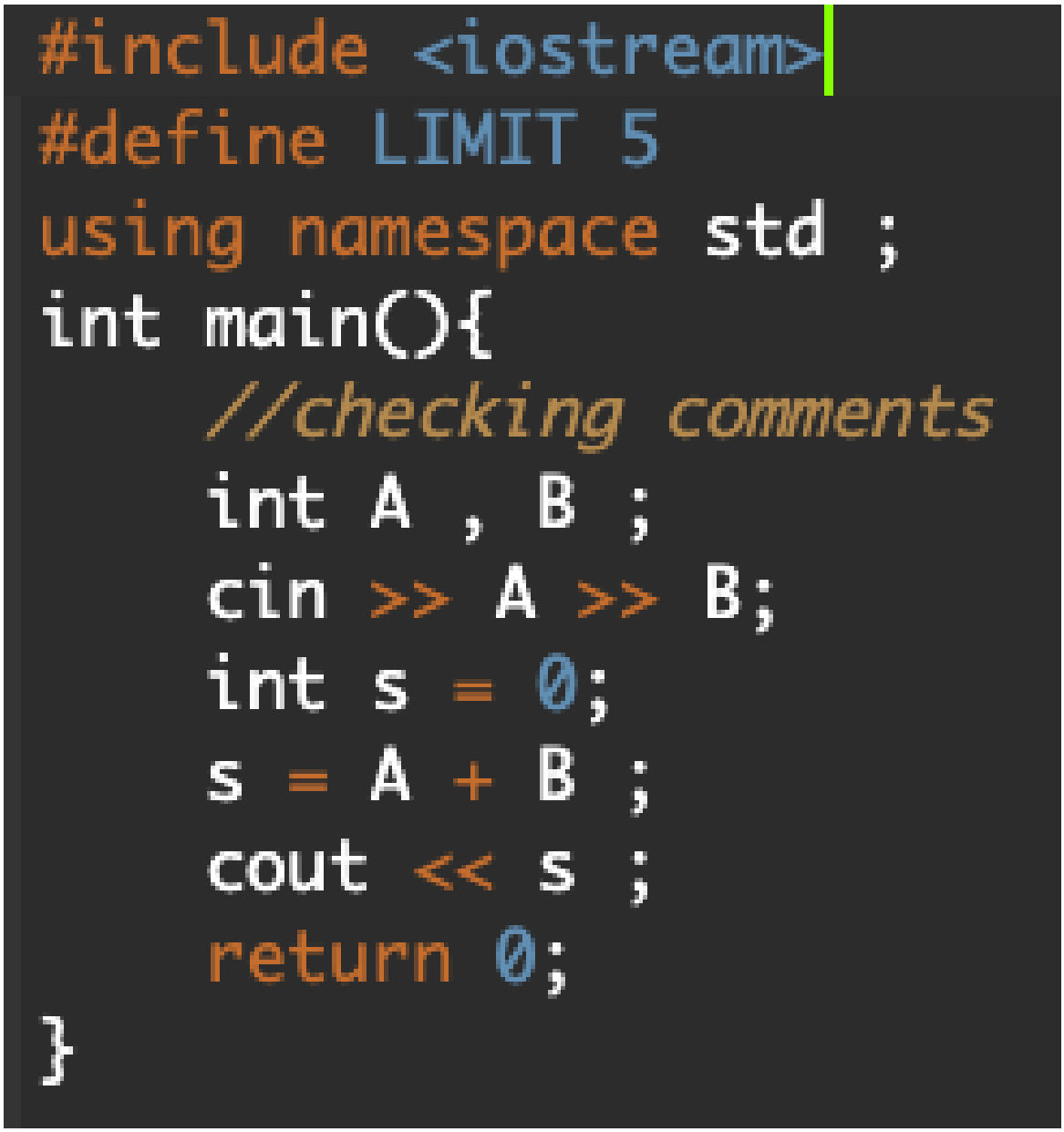
Corresponding output:



```
                                ENTER THE SOURCE CODE FILE NAME: SourceCode.txt
-------------------------------------------------------------------------------------------
        NUMBER              TOKEN                          PATTERN
-------------------------------------------------------------------------------------------


 Token   No :01  |            #  -------> |          Special Symbol
 Token   No :02  |      include  -------> |  Pre-Processor Directive
 Token   No :03  |            <  -------> |                Operator , LSHFT
 Token   No :04  |     iostream  -------> |                 Library
 Token   No :05  |            >  -------> |                Operator , RSHFT
 Token   No :06  |            #  -------> |          Special Symbol
 Token   No :07  |       define  -------> |  Pre-Processor Directive
 Token   No :08  |        LIMIT  -------> |                Variable ,  POINTER TO SYMBOL TABLE
 Token   No :09  |            5  -------> |                 Integer
 Token   No :10  |        using  -------> |                Keywords
 Token   No :11  |    namespace  -------> |                Keywords
 Token   No :12  |          std  -------> |                Keywords
 Token   No :13  |            ;  -------> |          Special Symbol
 Token   No :14  |          int  -------> |                Keywords
 Token   No :15  |         main  -------> |                Keywords
 Token   No :16  |            (  -------> |          Special Symbol
 Token   No :17  |            )  -------> |          Special Symbol
 Token   No :18  |            {  -------> |          Special Symbol
 Token   No :19  |          int  -------> |                Keywords
 Token   No :20  |            A  -------> |                Variable ,  POINTER TO SYMBOL TABLE
 Token   No :21  |            ,  -------> |          Special Symbol
 Token   No :22  |            B  -------> |                Variable ,  POINTER TO SYMBOL TABLE
 Token   No :23  |            ;  -------> |          Special Symbol
 Token   No :24  |          cin  -------> |                Keywords
 Token   No :25  |           >>  -------> |                Operator , INS
 Token   No :26  |            A  -------> |                Variable ,  POINTER TO SYMBOL TABLE
 Token   No :27  |           >>  -------> |                Operator , INS
 Token   No :28  |            B  -------> |                Variable ,  POINTER TO SYMBOL TABLE
 Token   No :29  |            ;  -------> |          Special Symbol
 Token   No :30  |          int  -------> |                Keywords
 Token   No :31  |            0  -------> |                 Integer
 Token   No :32  |            ;  -------> |          Special Symbol
 Token   No :33  |            A  -------> |                Variable ,  POINTER TO SYMBOL TABLE
 Token   No :34  |            +  -------> |                Operator , ADD
 Token   No :35  |            B  -------> |                Variable ,  POINTER TO SYMBOL TABLE
 Token   No :36  |            ;  -------> |          Special Symbol
 Token   No :37  |         cout  -------> |                Keywords
 Token   No :38  |           <<  -------> |                Operator , EXTR
 Token   No :39  |            ;  -------> |          Special Symbol
 Token   No :40  |            0  -------> |                 Integer
 Token   No :41  |            ;  -------> |          Special Symbol
 Token   No :42  |            }  -------> |          Special Symbol


TO CLOSE WINDOW
```

Source code 2 (code2.txt)

A c++ program to print a pattern using nested loop



```
#include <iostream>
#define LIMIT 5
using namespace std ;
int main(){
      //checking comments
      int A , B ;
      cin >> A >> B;
      int s = 0;
      s = A + B ;
      cout << s ;
      return 0;
}
```

Corresponding output:

```
                                ENTER THE SOURCE CODE FILE NAME: code2.txt
-------------------------------------------------------------------------------------------------
        NUMBER              TOKEN                          PATTERN
-------------------------------------------------------------------------------------------------


 Token   No :01  |            #  -------> |          Special Symbol
 Token   No :02  |      include  -------> |  Pre-Processor Directive
 Token   No :03  |            <  -------> |                 Operator , LSHFT
 Token   No :04  |          std  -------> |                 Keywords
 Token   No :05  |            +  -------> |                 Operator , ADD
 Token   No :06  |            +  -------> |                 Operator , ADD
 Token   No :07  |            >  -------> |                 Operator , RSHFT
 Token   No :08  |        using  -------> |                 Keywords
 Token   No :09  |    namespace  -------> |                 Keywords
 Token   No :10  |          std  -------> |                 Keywords
 Token   No :11  |            ;  -------> |           Special Symbol
 Token   No :12  |          int  -------> |                 Keywords
 Token   No :13  |         main  -------> |                 Keywords
 Token   No :14  |            (  -------> |           Special Symbol
 Token   No :15  |            )  -------> |           Special Symbol
 Token   No :16  |            {  -------> |           Special Symbol
 Token   No :17  |            (  -------> |           Special Symbol
 Token   No :18  |          int  -------> |                 Keywords
 Token   No :19  |            1  -------> |                  Integer
 Token   No :20  |            ;  -------> |           Special Symbol
 Token   No :21  |            <  -------> |                 Operator , LSHFT
 Token   No :22  |           11  -------> |                  Integer
 Token   No :23  |            ;  -------> |           Special Symbol
 Token   No :24  |            +  -------> |                 Operator , ADD
 Token   No :25  |            +  -------> |                 Operator , ADD
 Token   No :26  |            )  -------> |           Special Symbol
 Token   No :27  |            (  -------> |           Special Symbol
 Token   No :28  |          int  -------> |                 Keywords
 Token   No :29  |            1  -------> |                  Integer
 Token   No :30  |            ;  -------> |           Special Symbol
 Token   No :31  |            <  -------> |                 Operator , LSHFT
 Token   No :32  |            ;  -------> |           Special Symbol
 Token   No :33  |            +  -------> |                 Operator , ADD
 Token   No :34  |            +  -------> |                 Operator , ADD
 Token   No :35  |            )  -------> |           Special Symbol
 Token   No :36  |         cout  -------> |                 Keywords
 Token   No :37  |           <<  -------> |                 Operator , EXTR
 Token   No :38  |            *  -------> |                 Operator , MUL
 Token   No :39  |            ;  -------> |           Special Symbol
 Token   No :40  |         cout  -------> |                 Keywords
 Token   No :41  |           <<  -------> |                 Operator , EXTR
 Token   No :42  |            ;  -------> |           Special Symbol
 Token   No :43  |            0  -------> |                  Integer
 Token   No :44  |            ;  -------> |           Special Symbol
 Token   No :45  |            }  -------> |           Special Symbol
```