Lab Assignment

Name	S Vishwajith
Registration Number	23BCE1145
Subject	Compiler Design
Assignment Number	1

Aim:

To write a C++ program LA.cpp that acts as a lexical analyser to read a C++ source file Factorial.cpp containing a factorial program. The program should identify tokens such as keywords, identifiers, operators, constants, and symbols. It should then print each token with its type and value.

Algorithm:

1. Open the source file Factorial.cpp in read mode.
2. Repeat until End of File (EOF) is reached:
    a. Read each word (separated with a blank space) from the file.
    b. If the word has a special symbol, split it into 3 (before, special symbol, and after) and print each part as a token.
    c. Check if the word is a keyword, operator or a special symbol and print accordingly.
    d. If it's none of them, then the word is an identifier and print it accordingly.
3. Close the file after reading.

Codes:

Factorial.cpp:

```
#include <iostream>

using namespace std;

int factorial(int n)

{

  if (n <= 1)

    return 1;

  return n * factorial(n - 1);
```

```
}
int main()
{
   int n;
   cin >> n;
   cout << factorial(n) << endl;
   return 0;
}
```

LA.cpp:

```
#include <fstream>
#include <iostream>
#include <string>
#include <sstream>
#include <algorithm>
#include <vector>
using namespace std;
vector<string> keywords = {"include", "using", "namespace", "int", "if", "return"};
vector<string> operators = {"<", ">", "=", "-", ">>", "<<"};
vector<string> special_symbols = {"(", ")", ";", "{", "}", "#"};
bool hasPunctuator(string word)
{
   for (const auto &spec : special_symbols)
   {
      if (word.find(spec) != string::npos)
      {
         return true;
      }
   }
```

```
    return false;
}
void printToken(string word)
{
    if (word.length() > 1 && hasPunctuator(word))
    {
        for (const auto &spec : special_symbols)
        {
            size_t pos;
            while ((pos = word.find(spec)) != string::npos)
            {
                string beforePunct = word.substr(0, pos);
                if (!beforePunct.empty())
                {
                    printToken(beforePunct);
                }
                cout << "Type: SPECIAL SYMBOL, Value: " << spec << endl;
                word = word.substr(pos + spec.length());
            }
        }
        if (!word.empty())
        {
            printToken(word);
        }
        return;
    }
    if (word.find("<") != string::npos && word.find(">") != string::npos && word.length() > 1)
    {
```

```
        cout << "Type: OPERATOR, Value: <" << endl;
        word.erase(remove(word.begin(), word.end(), '<'), word.end());
        word.erase(remove(word.begin(), word.end(), '>'), word.end());
        printToken(word);
        cout << "Type: OPERATOR, Value: >" << endl;
        return;
    }
    auto itk = find(keywords.begin(), keywords.end(), word);
    auto ito = find(operators.begin(), operators.end(), word);
    auto itp = find(special_symbols.begin(), special_symbols.end(), word);
    if (itk != keywords.end())
    {
        cout << "Type: KEYWORD, Value: " << word << endl;
    }
    else if (ito != operators.end())
    {
        cout << "Type: OPERATOR, Value: " << word << endl;
    }
    else if (itp != special_symbols.end())
    {
        cout << "Type: PUNCTUATOR, Value: " << word << endl;
    }
    else
    {
        cout << "Type: IDENTIFIER, Value: " << word << endl;
    }
}
int main()
```

```
{
    ifstream inputFile("Factorial.cpp");
    if (!inputFile.is_open())
    {
        cerr << "Error: Could not open the file 'Factorial.cpp'" << endl;
        return 1;
    }
    string word;
    while (inputFile >> word)
    {
        printToken(word);
    }
    inputFile.close();
    return 0;
}
```

Output:

```
PS C:\Users\vishw\Coding\Compiler-Lab> cd "c:\Users\vishw\Coding\Compiler-Lab\Week-1\" ; if ($?) { g++ LA.cpp -o LA } ; if ($?) { .\LA }
Type: SPECIAL SYMBOL, Value: #
Type: KEYWORD, Value: include
Type: OPERATOR, Value: <
Type: IDENTIFIER, Value: iostream
Type: OPERATOR, Value: >
Type: KEYWORD, Value: using
Type: KEYWORD, Value: namespace
Type: IDENTIFIER, Value: std
Type: SPECIAL SYMBOL, Value: ;
Type: KEYWORD, Value: int
Type: IDENTIFIER, Value: factorial
Type: SPECIAL SYMBOL, Value: (
Type: KEYWORD, Value: int
Type: IDENTIFIER, Value: n
Type: SPECIAL SYMBOL, Value: )
Type: PUNCTUATOR, Value: {
Type: KEYWORD, Value: if
Type: SPECIAL SYMBOL, Value: (
Type: IDENTIFIER, Value: n
Type: IDENTIFIER, Value: <=
Type: IDENTIFIER, Value: 1
Type: SPECIAL SYMBOL, Value: )
Type: KEYWORD, Value: return
Type: IDENTIFIER, Value: 1
Type: SPECIAL SYMBOL, Value: ;
Type: KEYWORD, Value: return
Type: IDENTIFIER, Value: n
Type: IDENTIFIER, Value: *
Type: IDENTIFIER, Value: factorial
Type: SPECIAL SYMBOL, Value: (
Type: IDENTIFIER, Value: n
Type: OPERATOR, Value: -
Type: IDENTIFIER, Value: 1
Type: SPECIAL SYMBOL, Value: )
Type: SPECIAL SYMBOL, Value: ;
Type: PUNCTUATOR, Value: }
```

```
PS C:\Users\vishw\Coding\Compiler-Lab> cd "c:\Users\vishw\Coding\Compiler-Lab\Week-1\" ; if ($?) { g++ LA.cpp -o LA } ; if ($?) { .\LA }
Type: KEYWORD, Value: int
Type: IDENTIFIER, Value: main
Type: SPECIAL SYMBOL, Value: (
Type: SPECIAL SYMBOL, Value: )
Type: PUNCTUATOR, Value: {
Type: KEYWORD, Value: int
Type: IDENTIFIER, Value: n
Type: SPECIAL SYMBOL, Value: ;
Type: IDENTIFIER, Value: cin
Type: OPERATOR, Value: >>
Type: IDENTIFIER, Value: n
Type: SPECIAL SYMBOL, Value: ;
Type: IDENTIFIER, Value: cout
Type: OPERATOR, Value: <<
Type: IDENTIFIER, Value: factorial
Type: SPECIAL SYMBOL, Value: (
Type: IDENTIFIER, Value: n
Type: SPECIAL SYMBOL, Value: )
Type: OPERATOR, Value: <<
Type: IDENTIFIER, Value: endl
Type: SPECIAL SYMBOL, Value: ;
Type: KEYWORD, Value: return
Type: IDENTIFIER, Value: 0
Type: SPECIAL SYMBOL, Value: ;
Type: PUNCTUATOR, Value: }
PS C:\Users\vishw\Coding\Compiler-Lab\Week-1>
```

Type: SPECIAL SYMBOL, Value: #

Type: KEYWORD, Value: include

Type: OPERATOR, Value: <

Type: IDENTIFIER, Value: iostream

Type: OPERATOR, Value: >

Type: KEYWORD, Value: using

Type: KEYWORD, Value: namespace

Type: IDENTIFIER, Value: std

Type: SPECIAL SYMBOL, Value: ;

Type: KEYWORD, Value: int

Type: IDENTIFIER, Value: factorial

Type: SPECIAL SYMBOL, Value: (

Type: KEYWORD, Value: int

Type: IDENTIFIER, Value: n

Type: SPECIAL SYMBOL, Value:)

Type: PUNCTUATOR, Value: {

Type: KEYWORD, Value: if

Type: SPECIAL SYMBOL, Value: (

Type: IDENTIFIER, Value: n

Type: IDENTIFIER, Value: <=

Type: IDENTIFIER, Value: 1

Type: SPECIAL SYMBOL, Value:)

Type: KEYWORD, Value: return

Type: IDENTIFIER, Value: 1

Type: SPECIAL SYMBOL, Value: ;

Type: KEYWORD, Value: return

Type: IDENTIFIER, Value: n

Type: IDENTIFIER, Value: *

Type: IDENTIFIER, Value: factorial

Type: SPECIAL SYMBOL, Value: (

Type: IDENTIFIER, Value: n

Type: OPERATOR, Value: -

Type: IDENTIFIER, Value: 1

Type: SPECIAL SYMBOL, Value:)

Type: SPECIAL SYMBOL, Value: ;

Type: PUNCTUATOR, Value: }

Type: KEYWORD, Value: int

Type: IDENTIFIER, Value: main

Type: SPECIAL SYMBOL, Value: (

Type: SPECIAL SYMBOL, Value:)

Type: PUNCTUATOR, Value: {

Type: KEYWORD, Value: int

Type: IDENTIFIER, Value: n

Type: SPECIAL SYMBOL, Value: ;

Type: IDENTIFIER, Value: cin

Type: OPERATOR, Value: >>

Type: IDENTIFIER, Value: n

Type: SPECIAL SYMBOL, Value: ;

Type: IDENTIFIER, Value: cout

Type: OPERATOR, Value: <<

Type: IDENTIFIER, Value: factorial

Type: SPECIAL SYMBOL, Value: (

Type: IDENTIFIER, Value: n

Type: SPECIAL SYMBOL, Value:)

Type: OPERATOR, Value: <<

Type: IDENTIFIER, Value: endl

Type: SPECIAL SYMBOL, Value: ;

Type: KEYWORD, Value: return

Type: IDENTIFIER, Value: 0

Type: SPECIAL SYMBOL, Value: ;

Type: PUNCTUATOR, Value: }

Result:

The C++ program to perform lexical analysis of a factorial program was successfully run and the results were verified.