Compiler Design

Lexical Analysis

EXPERIMENT - 1

Roehit Ranganathan | RA1911033010017
1-7-2022

Aim:

To Build a program to perform as a lexical analyser of a compiler in C/C++/Java.

Program:

```
#include <bits/stdc++.h>
#include <regex>
using namespace std;
const char *specialsymbols[12] = {",", ".", ";", "{", "}", "(", ")", "[", "]",
"<<", ">>", ":"};
const char *operators[6] = {"+", "-", "*", "/", "?", "="};
const char *keyword[14] = {"const", "int", "char", "float", "main", "void",
"include", "return", "for", "while", "if", "else", "cin", "cout"};
void analyser(int n, vector<string> &arr, const char *parameter[], int &count,
string l)
{
    for (int i = 0; i < n; i++)
    {
        int size = arr.size();
        if (l.find(parameter[i]) != std::string::npos)
        {
            count++;
            if (std::find(arr.begin(), arr.end(), parameter[i]) != arr.end())
            {
                continue;
            }
            else
            {
                arr.push_back(parameter[i]);
            }
        }
    }
}
void print(vector<string> arr)
{
    for (int x = 0; x < arr.size(); x++)
    {
        cout << " " << arr[x] << " , ";
    }
}
bool iskeyword(string abc)
{
    int count = 0;
    for (int i = 0; i < 14; i++)
    {
        if (abc.find(keyword[i]) != std::string::npos)
        {
            count = 1;
            break;
```

```
        }
        else
            count = 0;
    }
    if (count == 1)
        return true;
    else
        return false;
}
bool isValidDelimiter(char ch)
{
    if ((!isdigit(ch) && !isalpha(ch)) || ch == '.' || ch == '#')
        return (true);
    else
        return (false);
}
bool isvalidIdentifier(char *str)
{
    if (isdigit(str[0]) || isValidDelimiter(str[0]) == true || str[0] == '\0')
        return (false);
    else
        return (true);
}
char *subString(char *str, int left, int right)
{
    int i;
    char *subStr = (char *)malloc(sizeof(char) * (right - left + 2));
    for (i = left; i <= right; i++)
        subStr[i - left] = str[i];
    subStr[right - left + 1] = '\0';
    return (subStr);
}
int main()
{
    fstream my_file;
    int count = 0;
    int count_keyword = 0;
    int count_operators = 0;
    int count_specialsymbols = 0;
    int count_identifiers = 0;
    char *line1;
    vector<string> arr_keywords;
    vector<string> arr_operators;
    vector<string> arr_specsymbols;
    vector<string> arr_identifiers;
    string line;
    string filename;
    cout << "Enter File Name to be analysed: ";
```

```
    cin >> filename;
    my_file.open(filename, ios::in);
    if (!my_file)
    {
        cout << "No such file";
    }
    else
    {
        cout << "Contents of file\n------------------------------" << endl;
        while (!my_file.eof())
        {
            getline(my_file, line);
            int n = line.length();
            char char_array[n + 1];
            strcpy(char_array, line.c_str());
            count++;
            cout << line << endl;
            if (line[0] != '#')
            {
                // KEYWORDS ANALYSER
                analyser(14, arr_keywords, keyword, count_keyword, line);
                int left = 0, right = 0;
                int l = line.length();
                // IDENTIFIER ANALYSER
                while (right <= l && left <= right)
                {
                    if (isValidDelimiter(line[right]) == false)
                    {
                        right++;
                    }
                    if (isValidDelimiter(line[right]) == true && left ==
right)
                    {
                        right++;
                        left = right;
                    }
                    else if (((isValidDelimiter(line[right]) == true) && (left
!= right)) || ((right == l) && (left != right)))
                    {
                        char *subStr = subString(char_array, left, right - 1);
                        if (isvalidIdentifier(subStr) == true &&
isValidDelimiter(line[right - 1]) == false && iskeyword(subStr) == false)
                        {
                            count_identifiers++;
                            if (std::find(arr_identifiers.begin(),
arr_identifiers.end(), subStr) == arr_identifiers.end())
                            {
                                arr_identifiers.push_back(subStr);
```

```
                            }
                        }
                        left = right;
                    }
                }
                // OPERATORS ANALYZER
                analyser(6, arr_operators, operators, count_operators, line);
                // SPECIAL SYMBOLS ANALYZER
                analyser(12, arr_specsymbols, specialsymbols,
count_specialsymbols, line);
            }
        }
        cout << "-------------------------------\nEnd Of File\n\nno of lines="
<< count - 1 << endl
             << "no of keywords=" << count_keyword << "\t\t{";
        print(arr_keywords);
        cout << "}\n"
             << "no of operators=" << count_operators << "\t\t{";

        print(arr_operators);
        cout << " }\n"
             << "no of special symbols=" << count_specialsymbols << "\t{";
        print(arr_specsymbols);
        cout << " }\nno of identifiers=" << count_identifiers << "\t\t{";
        print(arr_identifiers);
        cout
            << "}\n\n";
    }
    my_file.close();
    return 0;
}
```

Sample Text File:

#include <iostream.h>

int main(){

int a=1;

int b=2;

int c = a+b;

cout<<c;

return 0;

}

Result:

```
root@LAPTOP-26IF688U:/mnt/c/Users/ranga/Desktop# g++ -o exp1 exp1.cc
root@LAPTOP-26IF688U:/mnt/c/Users/ranga/Desktop# ./exp1
Enter File Name to be analysed: my_file.txt
Contents of file
------------------------------
#include <iostream.h>
int main(){
int a=1;
int b=2;
int c = a+b;
cout<<c;
return 0;
}

------------------------------
End Of File

no of lines=8
no of keywords=7                { int ,  main ,  cout ,  return , }
no of operators=4               { = ,  + ,  }
no of special symbols=10        { { ,  ( ,  ) ,  ; ,  << ,  } ,  }
no of identifiers=6             { a ,  b ,  c , }

root@LAPTOP-26IF688U:/mnt/c/Users/ranga/Desktop#
```