RA1911003010782
Harshit Aggarwal

Compile Design

11/1/22

EXP1: LEXICAL ANALYZER

AIM: To write a program to implement a lexical analyzer.

ALGORITHM:
1. Start.
2. Get the input program from the file code.txt.
3. Read the program line by line and check if each word in a line is a keyword, identifier, constant or an operator.
4. If the word read is an identifier, assign a number to the identifier and make an entry into the symbol table stored in code.txt.
5. For each lexeme read, generate a token as follows:
   - If the lexeme is an identifier, then the token generated is of the form <id, number>
   - If the lexeme is an operator, then the token generated is <op, operator>.
   - If the lexeme is a constant, then the token generated is <const, value>.
   - If the lexeme is a keyword, then the token is the keyword itself.
6. The stream of tokens generated are displayed in the console output.
7. Stop.

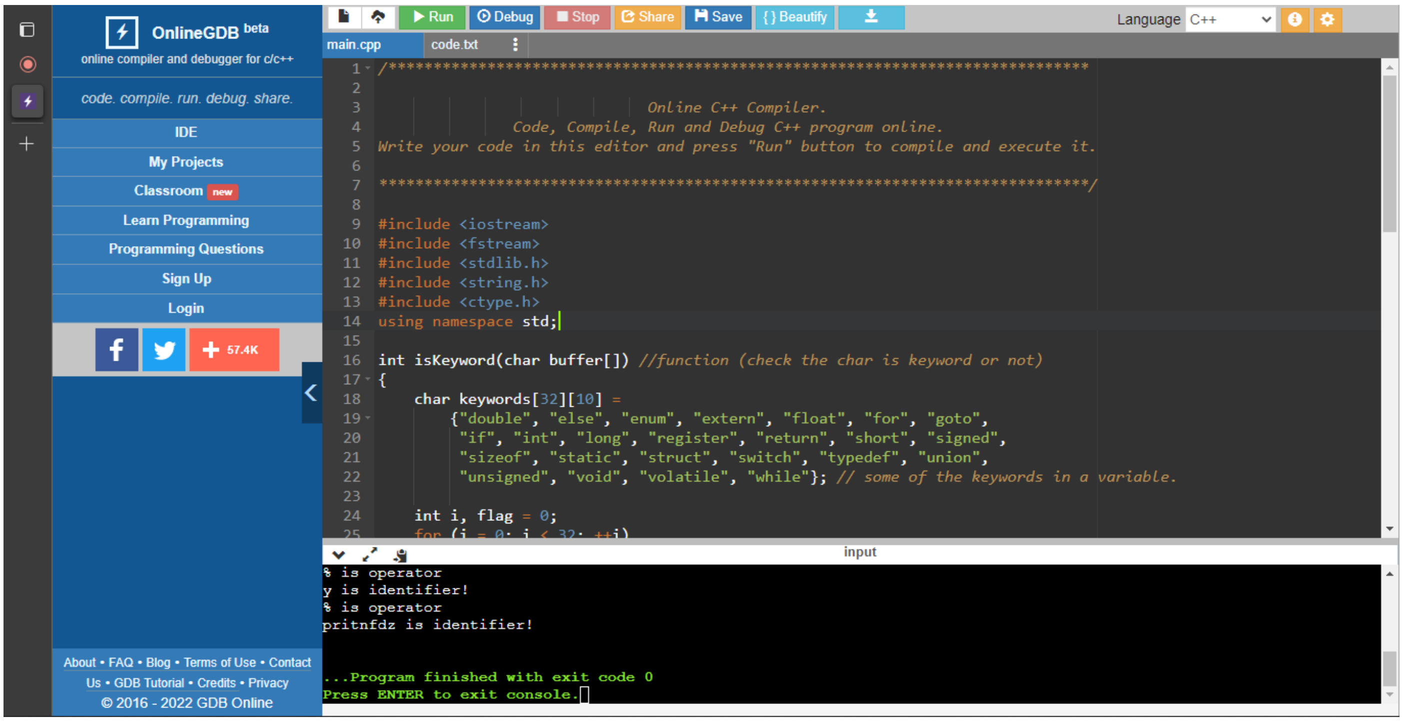
PROGRAM:

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
using namespace std;

int isKeyword(char buffer[]) //function (check the char is keyword or not)
{
   char keywords[32][10] =
      {"double", "else", "enum", "extern", "float", "for", "goto",
       "if", "int", "long", "register", "return", "short", "signed",
       "sizeof", "static", "struct", "switch", "typedef", "union",
       "unsigned", "void", "volatile", "while"}; // some of the keywords in a variable.

   int i, flag = 0;
   for (i = 0; i < 32; ++i)
   {
      if (strcmp(keywords[i], buffer) == 0)
      {
            flag = 1; // if we find any char set flag 1.
            break;
        }
    }
    return flag;
}
int main()
{
    char ch, buffer[15], operators[] = "+-*/%="; // operators and buffer declaration
    ifstream fin("code.txt");// reading input from the file
    int i, j = 0;
    if (!fin.is_open()) // No file found throw error.
    {
        cout << "error while opening the file\n";
        exit(0);
    }
    while (!fin.eof())
    {
        ch = fin.get();
        for (i = 0; i < 6; ++i)
        {
            if (ch == operators[i])
                cout << ch << " is operator\n"; //if the character is operator print "charcter is operator"
        }
        if (isalnum(ch))
        {
            buffer[j++] = ch;
        }
        else if ((ch == ' ' || ch == '\n') && (j != 0))
        {
            buffer[j] = '\0';
            j = 0;
            if (isKeyword(buffer) == 1)
                cout << buffer << " is keyword!\n"; // if flag is 1 print "given buffer is keyword"
            else
                cout << buffer << " is identifier!\n"; // else print "given buffer is identifier"
        }
    }
    fin.close(); //file close
    return 0;
}
```

INPUT:

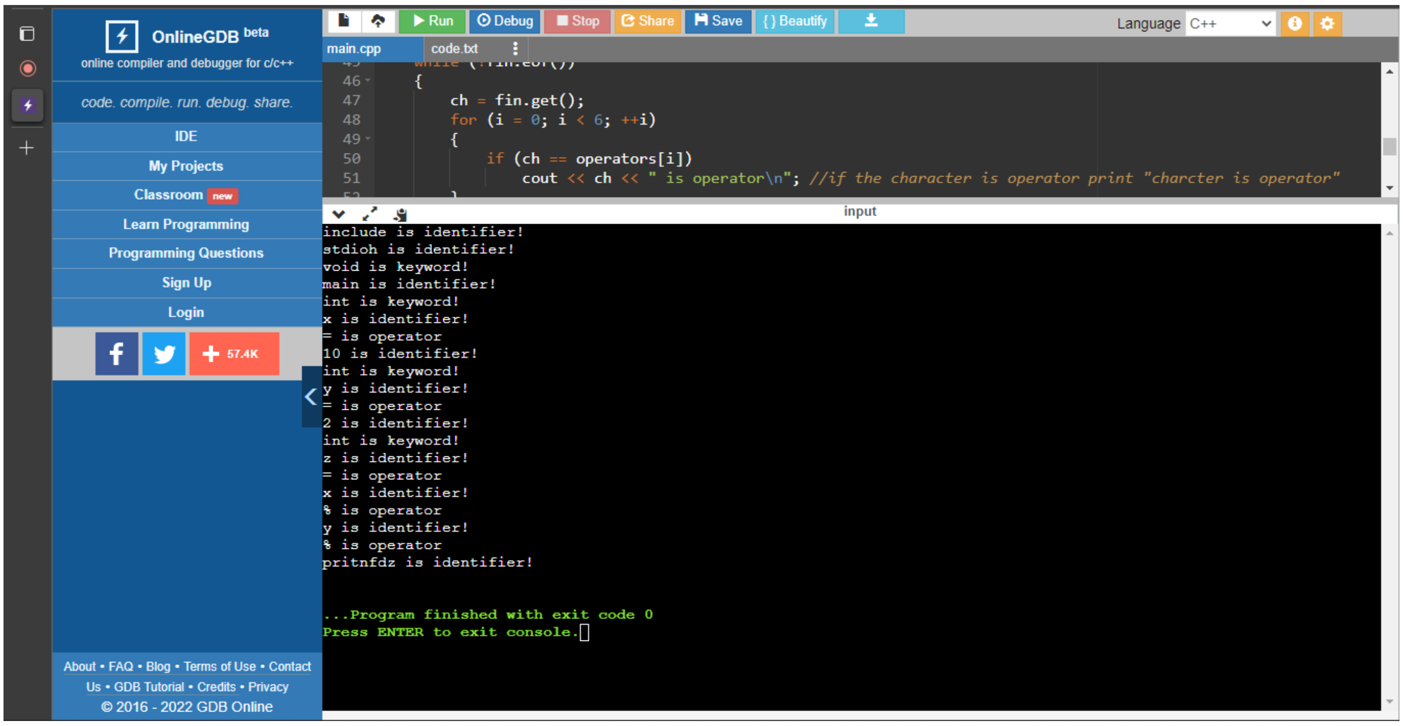
```
#include <stdio.h>
void main ( )
{
int x = 10;
int y = 2;
int z = x % y;
pritnf("%d",z);
}
```

OUTPUT:



```
/******************************************************************************

                              Online C++ Compiler.
               Code, Compile, Run and Debug C++ program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/

#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
using namespace std;

int isKeyword(char buffer[]) //function (check the char is keyword or not)
{
    char keywords[32][10] =
        {"double", "else", "enum", "extern", "float", "for", "goto",
         "if", "int", "long", "register", "return", "short", "signed",
         "sizeof", "static", "struct", "switch", "typedef", "union",
         "unsigned", "void", "volatile", "while"}; // some of the keywords in a variable.

    int i, flag = 0;
```

```
% is operator
y is identifier!
% is operator
pritnfdz is identifier!


...Program finished with exit code 0
Press ENTER to exit console.
```



```
    {
        ch = fin.get();
        for (i = 0; i < 6; ++i)
        {
            if (ch == operators[i])
                cout << ch << " is operator\n"; //if the character is operator print "charcter is operator"
```

```
include is identifier!
stdioh is identifier!
void is keyword!
main is identifier!
int is keyword!
x is identifier!
= is operator
10 is identifier!
int is keyword!
y is identifier!
= is operator
2 is identifier!
int is keyword!
z is identifier!
= is operator
x is identifier!
% is operator
y is identifier!
% is operator
pritnfdz is identifier!


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

The implementation of lexical analyser in C++ was compiled, executed and verified successfully.