University of Asia Pacific

Department of Computer Science & Engineering

Compiler Design Lab

CSE 430

Submitted to:

Baivab Das
Lecturer
CSE, University of Asia Pacific

Submitted by:

Asma Sultana
20101084
Section: B1

Problem

Write a program for lexical analysis that takes input from a file or keyboard and specifies each word or character into the tokens given below. The lexical analyzer should ignore redundant spaces, tabs, and newlines. It should also ignore comments and identify duplicate identifiers.

Sample Input (Console Input/ File Input):

```
void main()
{
int a, b, c;
//comment
int a = b*c + 10;
}
```

Sample output:

Keyword (2) : void, int
Identifier (4) : main, a, b, c
Arithmetic Operator (3) : =, *, +
Constant (1) : 10
Punctuation (2) : , , ;
Parenthesis (2) : { , }

Code

```
#include<bits/stdc++.h>
using namespace std;
set<string>key_string, iden_string,con_string , log_string;
set<char>arith_string, paren_string, pun_string;
int j ;
int isKeyword(char buffer[])
{
   char keywords[32][10] = {"auto","break","case","char","const","continue","default",
                            "do","double","else","enum","extern","float","for","goto",
                            "if","int","long","register","return","short","signed",
                            "sizeof","static","struct","switch","typedef","union",
                            "unsigned","void","volatile","while"
                           };
   char constant []= {'0','1','2','3','4','5','6','7','8','9'};
   int i, flag = 0;

   for(i = 0; i < 32; ++i)
   {
      if(strcmp(keywords[i], buffer) == 0)
      {
         flag = 1;
         break;
      }
   }
   /// constant.....................
   int constant_flag = 0 ;
   for(int i = 0 ; i < j ; i++)
   {
      constant_flag = 0 ;
      for(int k = 0 ; k < 10 ; k++)
      {
         if(buffer[i] == constant[k])
         {
```

```
            constant_flag = 1 ;
          }
        }
        if(constant_flag == 1)flag++;
    }
    for(int i = 0 ; i < j ; i++)
    {
        if(buffer[i]=='+'||buffer[i]=='-'||
            buffer[i]=='*'||buffer[i]=='/')
          flag = -1 ;
    }
    return flag;
}

int main()
{
    char ch, buffer[15], operators[] = "+-*/%=";
    char parenthesis[] = "(){}[]", pun[]= ",;:";
    ifstream fin("input.txt");
    int i;
    if(!fin.is_open())
    {
        cout<<"error while opening the file\n";
        exit(0);
    }
    int com_flag = 0 ,log_flag = 0 ;
    char comment , log = '$' ;
    while(!fin.eof())
    {
        ch = fin.get();
        /// comment .....
        if(ch == '/')
        {
            com_flag++;
            if(com_flag == 1)
```

```
    {
        comment = '/';
        continue;
    }
}
if(com_flag == 1)
{
    //cout << comment << "is operator\n";
    arith_string.insert(comment);
    comment =  NULL;
    com_flag = 0 ;
}
else if(com_flag == 2 && ch == '\n')
{
    com_flag = 0 ;
    comment = NULL;
    continue;
}
else if(com_flag == 2)continue;
/// logical op
if((ch == '>'||ch=='<'||ch=='='||ch=='!') && log=='$'){
    log_flag++;
    log = ch ;
    continue;
}
if(log_flag == 1 && (ch !='=')){
    log_flag = 0 ;
    log = '$';
}
if(log_flag == 1 && (ch =='=')){
   // cout<<log<<ch<<" is logical operator"<<'\n';
    string s = "";
    s +=log , s += ch ;
    log_string.insert(s);
    log_flag = 0 ;
```

```
    log = '$';
}
///.....
for(i = 0; i < 6; ++i)
{
    if(ch == parenthesis[i])
    {
       // cout<<ch<<" is parenthesis\n";
        paren_string.insert(ch);
    }
}
for(i = 0; i < 3; ++i)
{
    if(ch == pun[i])
    {
       // cout<<ch<<" is punctuation\n";
        pun_string.insert(ch);
    }
}
///Digit.....
int op_flag =  0 ;
for(i = 0; i < 6; ++i)
{
    if(ch == operators[i])
    {
       // cout<<ch<<" is operator\n";
        arith_string.insert(ch);
        op_flag = 1 ;
    }
}

if(isalnum(ch)||ch=='+'||ch=='-'||
        ch=='*'||ch=='/')
{
    buffer[j++] = ch;
```

```
    }
    else if((ch == ' ' || ch == '\n') && (j != 0))
    {
        buffer[j] = '\0';
        // cout << j << '\n';
        if(isKeyword(buffer) == 1)
        {
            //cout<<buffer<<" is keyword\n";
            key_string.insert(buffer);
        }
        else if(isKeyword(buffer) == j)
        {
            //cout << buffer<<" is constant\n";
            con_string.insert(buffer);
        }
        else if(isKeyword(buffer)== -1)
        {
            j = 0 ;
            continue;
        }
        else
        {
           // cout<<buffer<<" is indentifier\n";
            iden_string.insert(buffer);
        }

        j = 0;
    }

}
cout<<"keyword [ "<<key_string.size()<<"] : ";
for(auto u : key_string)cout<<u<<" ";
cout<<'\n';
cout<<"Identifier [ "<<iden_string.size()<<"] : ";
for(auto u : iden_string)cout<<u<<" ";
```

```
    cout<<'\n';
    cout<<"Arithmetic operator [ "<<arith_string.size()<<"] : ";
    for(auto u : arith_string)cout<<u<<" ";
    cout<<'\n';
    cout<<"logical operator [ "<<log_string.size()<<"] : ";
    for(auto u : log_string)cout<<u<<" ";
    cout<<'\n';
    cout<<"Constant [ "<<con_string.size()<<"] : ";
    for(auto u : con_string)cout<<u<<" ";
    cout<<'\n';
    cout<<"Punctuation [ "<<pun_string.size()<<"] : ";
    for(auto u : pun_string)cout<<u<<" ";
    cout<<'\n';
    cout<<"Parenthesis [ "<<paren_string.size()<<"] : ";
    for(auto u : paren_string)cout<<u<<" ";
    cout<<'\n';
    fin.close();

    return 0;
}
```

here is the observed output: