|  |
| --- |
| **Lab 7: Token Extraction and Context Free Grammar (CFG)** |

**Exercise 1**

Read the token.txt file generated in result of lexical analyzer. Create a structure of token and fetch all tokens from token.txt file into the structure array. Before reading the file make sure you have inserted end of file token in token.txt.

**Source Code:**

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

struct token{

string cp,vp,lno;

};

int main(){

ifstream infile;

string str,str1,t1;

int s,a=0,b=0;

//To read the number of lines from file we open a file twice:

infile.open("lab.txt");

//Check for error

if(infile.fail()){

cerr<<"Error in opening file"<<endl;

exit(1);

}

while(!infile.eof()){

infile>>str;

s++;

} str.clear();

cout<<"number=:"<<s<<endl;

infile.close();

token tok[s];

infile.open("lab.txt");

//Check for error

if(infile.fail()){

cerr<<"Error in opening file"<<endl;

exit(1);

}

int cc=0,c=0;

while(getline(infile,str)){

str1+=str;

for(int i=0;i<str1.length();i++)

{

if(i==0) continue;

if(str1[i]==',' && b==0) ( tok[a].cp=t1 );

else if(str1[i]==',' && b==1) ( tok[a].vp=t1 );

else if(str1[i]==')' && b==2) ( tok[a].lno=t1 );

else ( t1+=str1[i] );

if(str[i]==','){

b++;

t1.clear();

}

cc++;

}

c++;

b=0;

a++;

str1.clear();

t1.clear();

}

infile.close();

cout<<" CP VP LINE\_NO\n"<<endl;

for(int i=0;i<a;i++){

cout<<i<<": "<<tok[i].cp<<" "<<tok[i].vp<<" "<<tok[i].lno<<endl;

}

return 0;

}

**SCREENSHOT:**

