**Week 3 Lab Assignment: Lexical Analyzer**

**Session: 2021 – 2025**

**Submitted by:**

Ghulam Mustafa (2021-CS-39)

**Supervised by:**

Sir. Laeeq Khan Niazi

Department of Computer Science

**University of Engineering and Technology Lahore Pakistan**

**Rules Defined:**

1. Function to check if the the string is a word
2. Function to check if the string is and operator
3. Function to check if the string is a punctuator
4. Function to check if the string is an identifier
5. Function to check if the string is a number
6. A tokernize function convert the code into different tokens, it works by checking the boundries using a space or identifieng a punctuation

**Code:**

#include <iostream>

#include <sstream>

#include <cctype>

#include <vector>

using namespace std;

bool isKeyword(string token){

string keywords[]={"int","float","bool","if","else","while","do","for","return","void"};

for(int i=0;i<10;i++){

if(token==keywords[i]){

return true;

}

}

return false;

}

bool isOperator(string token){

string operators[]={"+","-","\*","=",">","<","<=",">=","==","!="};

for(int i=0;i<10;i++){

if(token==operators[i]){

return true;

}

}

return false;

}

bool isPunctuator(string token){

string punctuators[]={",","{","}","(",";",")"};

for(int i=0;i<6;i++){

if(token==punctuators[i]){

return true;

}

}

return false;

}

bool isIdentifier(string token){

if((token[0]>='a' && token[0]<='z')||(token[0]>='A' && token[0]<='Z')||token[0]=='\_'){

return true;

}

return false;

}

bool isNumber(string token){

for(char c:token){

if(!isdigit(c)){

return false;

}

}

return true;

}

void tokenize(const string& str, vector<string>& tokens){

string token;

for(char c:str){

if(isspace(c)||ispunct(c)){

if(!token.empty()){

tokens.push\_back(token);

token.clear();

}

if(ispunct(c)){

tokens.push\_back(string(1,c));

}

} else{

token+=c;

}

}

if(!token.empty()){

tokens.push\_back(token);

}

}

int main(){

string test\_code2="main(){\nint a=10;\n}";

string test\_code="for(int i=0;i<6;i++){\nif(token==punctuators[i]){\nreturn true;\n}\n}";

cout<<"This is the test code : "<<test\_code<<"\n";

vector<string> tokens;

tokenize(test\_code,tokens);

cout<<"Results: \n";

int count=0;

for(const string& token:tokens){

if(isKeyword(token)){

cout<<token<<" keyword"<<endl;

count++;

}

else if(isIdentifier(token)){

cout<<token<<" identifier"<<endl;

count++;

}

else if(isOperator(token)){

cout<<token<<" operator"<<endl;

count++;

}

else if(isPunctuator(token)){

cout<<token<<" punctuator"<<endl;

count++;

}

else if(isNumber(token)){

cout<<token<<" number"<<endl;

count++;

}

else{

cout<<token<<" is unknown"<<endl;

count++;

}

}

cout<<"Total tokens: "<<count<<endl;

return 0;

}

**Screenshots of Outputs:**

**Test code 1 Results:**

In figure 1 you can identify that the code ran successfully, and it identified all the tokens successfully.

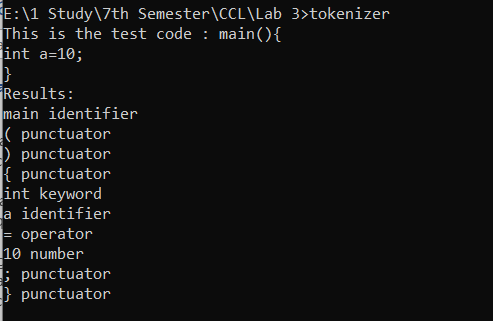


Figure 1

**Test code 2 Results:**

In figure 2, you can see the results of the second test, I tried to store the code without \n and added break lines, as it is against the rules of C++ therefore, I got lot of errors.

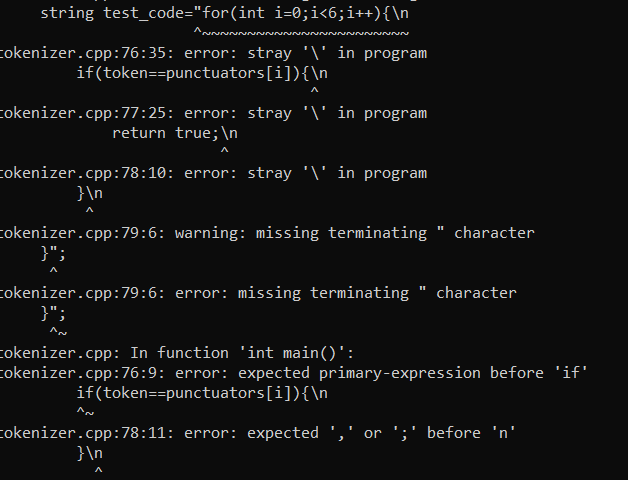


Figure 2

**Test code 3 Results:**

You can see the from the figure 3, the same code I from the figure 2 I tried to run it by adding \n, instead of directly breaking the lines the code ran successfully.

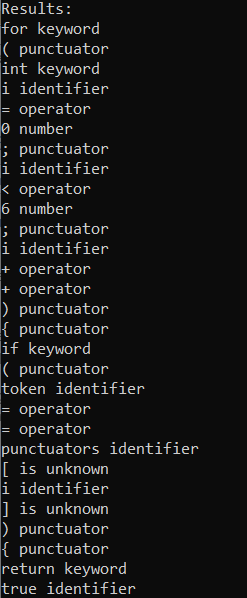


Figure 3