182 lines (174 sloc) | 6.12 KB

```
#include<iostream>
#include<regex>
#include<vector>
#include<fstream>
#include<string>

using namespace std;

string stripL(string );
string stripR(string );
string strip(string );
string getSeperatorName(string );
string getType(string );

const string separators = "[=<>\\(\\)\\{\\},\\.;\\[\\]]|(\\s)+";
const string keywords = "(auto)|(break)|(case)|(char)|(const)|(continue)|(default)|(do)|(double
const string number = "0|(-?[1-9][0-9]*)";
const string lattersValidName = "[A-Za-z_][A-Za-z0-9_]*";

int main(){
    //freopen("output.txt","w",stdout);
    /*if(regex_match("=",regex(separators))){
        cout<<"match"<<endl;
    }else{
        cout<<"Not "<<endl;
    }*/
    string fileName="sourceCode.c";
    ifstream fs;
    fs.open(fileName);
    if (!fs) {
        cout << "Unable to open file";
        exit(1); // terminate with error
    }else{
        cout<<"File open success."<<endl;
    }
    bool multilineCommentOut{false};
    string line;
    smatch match;
    vector<pair<string,string>> tokens;
    while(getline(fs,line)){
        line = strip(line);
        //cout<<"After strip line is : \""<<line<<"\""<<endl;
        ///Preprocessor
        if(line[0]=='#'){
            tokens.push_back({"PREPROCESSOR","#"});

            line = line.substr(1);
            if(regex_search(line,match,regex("<|(\\s)+"))){
```

```
48                  tokens.push_back({"PREPROCESSOR_TYPE",match.prefix()});
49                  tokens.push_back({getSeperatorName(match.str()),match.str()});
50                  line=match.suffix();
51              }
52              if(regex_search(line,match,regex(">|(\\s)+"))){
53                  if(match.str()==">"){
54                      tokens.push_back({"HEADER_FILE",match.prefix()});
55                      tokens.push_back({getSeperatorName(match.str()),match.str()});
56                  }else{
57                      tokens.push_back({"CONSTANT_IDENTIFER",match.prefix()});
58                      tokens.push_back({getSeperatorName(match.str()),match.str()});
59                      tokens.push_back({"CONSTANT_VALUE",match.suffix()});
60                  }
61              }continue;
62          }
63          start:
64          ///Skipping line checking or Comment out part
65          if(multilineCommentOut){
66              //cout<<"commented"<<endl;
67              if(regex_search(line,match,regex("\\*/"))){
68                  //cout<<"Finished comment"<<endl;
69                  line=match.suffix();
70                  multilineCommentOut=false;
71                  goto start;
72              }continue;
73
74          }else if(line==""||regex_match(line,regex("^//.*"))){
75              //cout<<"empty line or comment line"<<endl;
76              continue;
77          }
78          else if(regex_search(line,match,regex("//"))){
79              //cout<<"sub comment line"<<endl;
80              line = strip(match.prefix());
81              //cout<<"Remaining line is : \""<<line<<"\""<<endl;
82          }else if(regex_search(line,match,regex("/\\*"))){
83              //cout<<"MultiLine comment out found"<<endl;
84              line = match.prefix();
85              multilineCommentOut=true;
86              string commentString=match.suffix();
87              if(regex_search(commentString,match,regex("\\*/"))){
88                  //cout<<"Finish in same line"<<endl;
89                  line+=match.suffix();
90                  multilineCommentOut=false;
91                  goto start;
92              }
93              if(line==""){
94                  continue;
95              }
96          }
97          //cout<<"Processing line is : \""<<line<<"\""<<endl;
98          if(line==""){
99              //cout<<"Empty line"<<endl;
```

```
            continue;
        }
        ///Separator finding
        if(regex_search(line,match,regex(separators))){
            //cout<<"Separator found : \""<<match.str()<<"\""<<endl;
            //cout<<"Name is : "<<getSeperatorName(match.str())<<endl;
            if(match.prefix()!=""){
                //cout<<"prefix found : \""<<match.prefix()<<"\""<<endl;
                //cout<<"Name is : "<<getType(match.prefix())<<endl;
                tokens.push_back(make_pair(getType(match.prefix()),match.prefix()));
            }
            tokens.push_back(make_pair(getSeperatorName(match.str()),match.str()));
            line=match.suffix();
            goto start;
        }
    }
    cout<<endl<<"********* Tokens are **********"<<endl<<endl;
    for(pair<string,string> token : tokens){
        cout<<" <'"<<token.first<<"', '"<<token.second<<"'> "<<endl;
    }cout<<endl;

}

string getSeperatorName(string separator){
    if(separator=="("){
        return "OPEN_PARANTHESES";
    }else if(separator==")"){
        return "CLOSE_PARANTHESES";
    }else if(separator=="{"){
        return "OPEN_CURLY_BRACES";
    }else if(separator=="}"){
        return "CLOSE_CURLY_BRACES";
    }else if(separator=="["){
        return "OPEN_SQUARE_BRAKET";
    }else if(separator=="]"){
        return "CLOSE_SQUARE_BRAKET";
    }else if(separator=="<"){
        return "OPEN_ANGULAR_BRAKET";
    }else if(separator==">"){
        return "CLOSE_ANGULAR_BRAKET";
    }else if(separator==","){
        return "COMA_DELIMATOR";
    }else if(separator=="."){
        return "DOT_OPERATOR";
    }else if(separator==";"){
        return "SEMECLONE";
    }else if(separator=="="){
        return "ASSIGNMENT_OPERATOR";
    }else{
        return "SPACES";
    }
}
```

```
string getType(string str){
    if(regex_match(str,regex(keywords))){
        return "KEYWORD";
    }else if(regex_match(str,regex(lattersValidName))){
        return "IDENTIFIRE";
    }else if(regex_match(str,regex(number))){
        return "CONSTANT_NUMBER";
    }else{
        return "UNKNOWN";
    }
}

string stripL(string input_str){
    int starting_pointer=0;
    while(input_str[starting_pointer]==' '){
        starting_pointer++;
    }
    return input_str.substr(starting_pointer);
}
string stripR(string input_str){
    int ending_pointer=input_str.size()-1;
    while(input_str[ending_pointer]==' '){
        ending_pointer--;
    }
    return input_str.substr(0,ending_pointer+1);
}
string strip(string input_str){
    string part = stripL(input_str);
    part = stripR(part);
    return part;
}
```