```
#include<iostream>
#include<regex>
#include<vector>
#include<fstream>
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)|(else)|(enum)|(extern)|(fl
oat)|(for)|(goto)|(if)|(int)|(long)|(register)|(return)|(short)|(signed)|(sizeof)|(static)|(struct)|(swit
ch)|(typedef)|(union)|(unsigned)|(void)|(volatile)|(while)";
const string number = "0|(-?[1-9][0-9]*)";
const string lattersValidName = "[A-Za-z_][A-Za-z0-9_]*";
int main(){
   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);
      ///Preprocessor
      if(line[0]=='#'){
         tokens.push_back({"PREPROCESSOR","#"});
         line = line.substr(1);
         if(regex_search(line,match,regex("<|(\\s)+"))){
            tokens.push_back({"PREPROCESSOR_TYPE",match.prefix()});
            tokens.push_back({getSeperatorName(match.str()),match.str()});
            line=match.suffix();
         }
         if(regex_search(line,match,regex(">|(\\s)+"))){
            if(match.str()==">"){
               tokens.push_back({"HEADER_FILE",match.prefix()});
               tokens.push_back({getSeperatorName(match.str()),match.str()});
            }else{
               tokens.push_back({"CONSTANT_IDENTIFER",match.prefix()});
               tokens.push_back({getSeperatorName(match.str()),match.str()});
               tokens.push_back({"CONSTANT_VALUE",match.suffix()});
            }
        }continue;
    }
    start:
    ///Skipping line checking or Comment out part
    if(multilineCommentOut){
        if(regex_search(line,match,regex("\\*/"))){
            line=match.suffix();
            multilineCommentOut=false;
            goto start;
        }continue;
    }else if(line==""||regex_match(line,regex("^//.*"))){    continue; }
    else if(regex_search(line,match,regex("//"))){
        line = strip(match.prefix());
    }else if(regex_search(line,match,regex("/\\*"))){
        line = match.prefix();
        multilineCommentOut=true;
        string commentString=match.suffix();
        if(regex_search(commentString,match,regex("\\*/"))){
            line+=match.suffix();
            multilineCommentOut=false;
            goto start;
        }if(line==""){    continue; }
    }if(line==""){    continue; }
    ///Separator finding
    if(regex_search(line,match,regex(separators))){
        if(match.prefix()!=""){
            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;
}
```