#include<iostream>

#include<regex>

#include<vector>

#include<fstream>

using namespace std;

string stripL(string );

string stripR(string );

string strip(string );

string get\_subString(string ,char ,char );

void error\_msg\_show(int line,string message){

cout<<"Error found in line : "<<line<<endl;

cout<<"Error message : "<<message<<endl<<endl;

}

int main(){

regex regex\_header\_start("(#\\s\*include\\s\*)(.\*)");

regex regex\_header\_line\_valid("(#\\s\*include\\s\*<)(.\*)>");

regex regex\_main("(.\*) main(.\*)");

regex regex\_main\_parm("(.\*) main()(.\*)");

regex regex\_main\_return\_type\_void("\\s\*void(.\*)");

regex regex\_main\_return\_type\_int("\\s\*int(.\*)");

regex regex\_return\_statment("(.\*)return\\s(.\*)");

regex regex\_return\_int("(.\*)return\\s\*(0|-?([1-9][0-9]\*)\\s\*);");

regex regex\_digit("[0-9]");

regex regex\_digits("([1-9][0-9]\*)");

regex regex\_scop\_start("(.\*)\\{(.\*)");

regex regex\_scop\_end("(.\*)\\}(.\*)");

string fileName="001source\_code.c";

ifstream fs;

fs.open(fileName);

if (!fs) {

cout << "Unable to open file";

exit(1);

}

bool multilineCommentOut{false};

bool return\_need{false};

bool scop\_start{false};

int lineNumber=1;

string line;

smatch match;

while(getline(fs,line)){

start:

line = strip(line);

cout<<"After strip line is : \""<<line<<"\""<<endl;

///Skiping line checking

if(multilineCommentOut){

cout<<"Line commented"<<endl;

if(regex\_search(line,match,regex("\\\*/"))){

cout<<"Finished comment"<<endl;

line=match.suffix();

multilineCommentOut=false;

goto start;

}continue;

}else if(line==""||regex\_match(line,regex("^//.\*"))){

cout<<"empty line or comment line"<<endl;

continue;

}

else if(regex\_search(line,match,regex("//"))){

cout<<"sub comment line"<<endl;

line = strip(match.prefix());

cout<<"Remaining line is : \""<<line<<"\""<<endl;

}else if(regex\_search(line,match,regex("/\\\*"))){

cout<<"MultiLine comment out found"<<endl;

line = match.prefix();

multilineCommentOut=true;

string commentString=match.suffix();

if(regex\_search(commentString,match,regex("\\\*/"))){

cout<<"Finish in same line"<<endl;

line+=match.suffix();

multilineCommentOut=false;

goto start;

}

if(line==""){

continue;

}

}

cout<<"Processing line is : \""<<line<<"\""<<endl;

bool isvalidLine=true;

if(regex\_match(line,regex\_scop\_start)){

scop\_start=true;

}

if(regex\_match(line,regex\_header\_start)){

cout<<"Header found"<<endl;

if(regex\_match(line,regex\_header\_line\_valid)){

cout<<"A Valid header found"<<endl;

string header = get\_subString(line,'<','>');

header=strip(header);

cout<<"Header is :\""<<header<<"\""<<endl;

if(header!="stdio.h"){

error\_msg\_show(lineNumber,header+" not defined header.");

return 0;

}

}else{

error\_msg\_show(lineNumber,"Invalid header diclaration.");

return 0;

}

}else if(regex\_match(line,regex\_main)){

cout<<"Main found."<<endl;

if(regex\_match(line,regex\_main\_parm)){

cout<<"Main function found."<<endl;

if(regex\_match(line,regex\_main\_return\_type\_int)){

cout<<"Main should be return an integer."<<endl;

return\_need = true;

}else if(regex\_match(line,regex\_main\_return\_type\_void)){

cout<<"Main return not need."<<endl;

}else{

error\_msg\_show(lineNumber,"Invalid return type of Main function.");

return 0;

}

}else{

error\_msg\_show(lineNumber,"Invalid main function declaration.");

return 0;

}

}

else if(regex\_match(line,regex\_return\_statment)){

cout<<"Return statement found"<<endl;

if(!scop\_start){

error\_msg\_show(lineNumber,"Not in a scope.");

return 0;

}if(regex\_match(line,regex\_return\_int)){

cout<<"Return type an integer found."<<endl;

return\_need = false;

}else{

error\_msg\_show(lineNumber,"Return type not an integer.");

return 0;

}

}else{

isvalidLine = false;

}

if(regex\_match(line,regex\_scop\_end)){

scop\_start=false;

isvalidLine = true;

}

if(!isvalidLine){

error\_msg\_show(lineNumber,"Invalid statement found.");

return 0;

}

lineNumber++;

}

fs.close();

if(return\_need){

error\_msg\_show(lineNumber,"Return not found in the scope.");

return 0;

}

if(scop\_start){

error\_msg\_show(lineNumber,"Scope not finished.");

return 0;

}

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*Compilation success.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

return 0;

}

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;

}

string get\_subString(string str,char start\_dilam,char end\_dilam){

stringstream ss(str);

string part;

getline(ss,part,end\_dilam);

ss = stringstream(part);

getline(ss,part,start\_dilam);

getline(ss,part,start\_dilam);

return part;

}