**语言词法分析器实验报告**

一）**实验目的：**

## （1）理解词法分析在编译程序中的作用：其主要任务是从左到右逐个字符的对 源程序进行扫描，按照源语言的语法规则识别出一个个的单词符号，把识别出来 的单词符号存入符号表中，并产生用于词法分析的记号序列。

（2）加深对有穷自动机模型的理解：语法分析的过程就是一个对所碰到的单词 符号进行自动机模拟的过程，碰到一个自己需要的字符就进行状态转换，并最终 到达终止状态，即完成了一个运用源语言进行分析的过程。

（3）掌握词法分析程序的实现方法和技术：利用 c 语言描述了一个词法分析器， 旨在对于词法分析的过程站在计算机的角度进行了解与体会。

（4）用 c 语言对一个简单语言的子集编制一个一遍扫描的编译程序，以加深对 编译原理的理解,掌握编译程序的实现方法和技术。

（5）在课堂上比较详细的了解了LEX之后，自己回来又查找了相关资料，实现了使用LEX进行词法分析器构建的过程。LEX简单易懂，代码也很短。

**二） 实验环境**

## Windows 10操作系统；

本次实验的C++代码实现依附于codeblock环境。Lex实现依附与Sublime text3、Flex程序以及终端命令行。

**三） 实验过程：**

## **1、词法分析过程**：

## 根据输入单词的第一个有效字符（在相关符号与标示符判断的时候还需要预读一个进行判定），判断单词类，产生类号。对于标识符和常数，需分别与标识 符表和常数表中已登记的元素相比较，比较之后输出结果。

Main 函数中设计若干计数器，用于，语句，单词数，字符个数的计数，并且在主函数中设定一个 id 的全局变量进行标示符 id 的统计。

根据课本上的相关描述，写出whitespace,id,num,comment,note 等相关的正则表达式（whitespace表示空白符号，如空格与tab。Id 表示正确的标示符，num表示正确的数字，包扩整形与浮点型。Comment 代表注释，note表示printf中的相关输出。）

其余的符号和关键字，因为数量巨大，换做让程序自己进行选择的方式来实现。

**2．实验过程中的问题**：

刚开始的时候比较迷惘，所以选择了仔细阅读课本，在看过几遍课本提供的Pascal之后逐渐明白了相关方面的一些细节。在大概完成代码之后，试着分析了一个小程序，有部分差别。一步一步优化，一点一点考虑，功夫不负有心人啦。一步一步debug改掉了一些细节上的漏洞。大功告成。

关于LEX呢，以前完全没有接触过，听周围人说不错，本来写好了C++就不准备动手弄 lex 了，但是觉得挺有趣就来试试，在windows配置许久才终于搞定环境变量的问题。试着写了几个简单的LEX小程序之后，觉得稍微对这门新语言有些掌握后，就开始着手准备编写作业。完成之后，一看真心感觉不错，就对于词法分析这一项功能来说，比C++实现优秀许多。无论在方便程度还是效率方面，真心强悍于一般C++。希望以后的相关实践中可以借助LEX来做一些有用并且惊艳的事情。

**3、实验测试代码：**这是测试用代码**：** #include<stdio.h>

int main()

{

printf("Hello world!\n");

return 0;

}

**4．C++实现代码**

//

// 词法分析.cpp

// Main

// Created by 吴浩男 on 15/10/30.

// Copyright © 2015年 吴浩男. All rights reserved.

// Mr.Phoebe的专栏 http://blog.csdn.net/u013007900

#include<bits/stdc++.h>

using namespace std;

set<string> S[8+5];

vector<pair<int,int> > error;

void output(int type,string &buffer)//将buffer的[0..len-2]推入S[type]这个集合里

{

int Size=buffer.size();

S[type].insert(buffer.substr(0,Size-1));

buffer.clear();

}

int main()

{

freopen("data.txt","r",stdin);

string line;

int row=0,col=0;

for(int i=0; i<8; i++)S[i].clear(); //对应的每个符号的单词数组

int state=0;

string buffer;

while(getline(cin,line))

{

line+='\n';

row++;

col=0;

if(state!=10)state=0;

for(int i=0; i<line.size(); i++)

{

col++;

buffer+=line[i];

if(state==0)

{

if(isalpha(line[i]))state=1;

else if(isdigit(line[i]))state=2;

else if(line[i]=='<'||line[i]=='>'||line[i]=='='||line[i]=='!')state=8;

else if(line[i]=='/')state=9;

else if(line[i]=='+'||line[i]=='-'||line[i]=='\*'||line[i]=='('||line[i]==')'||line[i]==';'||line[i]=='\'')state=13;

else if(line[i]!='\n')

{

error.push\_back(make\_pair(row,col));

state=0;

buffer.clear();

}

continue;

}

if(state==1)

{

if(isalpha(line[i])||isdigit(line[i]))state=1;

else

{

output(0,buffer);

state=0;

i--;

col--;

}

continue;

}

if(state==2)

{

if(isdigit(line[i]))state=2;

else if(line[i]=='.')state=3;

else if(line[i]=='E')state=5;

else

{

output(1,buffer);

state=0;

i--;

col--;

}

continue;

}

if(state==3)

{

if(isdigit(line[i]))state=4;

else

{

error.push\_back(make\_pair(row,col));

state=0;

buffer.clear();

}

continue;

}

if(state==4)

{

if(isdigit(line[i]))state=4;

else

{

output(1,buffer);

state=0;

i--;

col--;

}

continue;

}

if(state==5)

{

if(isdigit(line[i]))state=7;

else if(line[i]=='+'||line[i]=='-')state=6;

else

{

error.push\_back(make\_pair(row,col));

state=0;

buffer.clear();

}

continue;

}

if(state==6)

{

if(isdigit(line[i]))state=7;

else

{

error.push\_back(make\_pair(row,col));

state=0;

buffer.clear();

}

continue;

}

if(state==7)

{

if(isdigit(line[i]))state=7;

else

{

output(1,buffer);

state=0;

i--;

col--;

}

continue;

}

if(state==8)

{

if(line[i]=='=')state=14;

else

{

if(buffer[0]=='<'||buffer[0]=='>') output(2,buffer);

else output(5,buffer);

state=0;

i--;

col--;

}

continue;

}

if(state==9)

{

if(line[i]=='\*')state=10;

else

{

output(3,buffer);

state=0;

i--;

col--;

}

continue;

}

if(state==10)

{

if(line[i]=='\*') state=11;

else state=10;

continue;

}

if(state==11)

{

if(line[i]=='/') state=12;

else state=10;

continue;

}

if(state==12)

{

output(4,buffer);

state=0;

i--;

col--;

continue;

}

if(state==13)

{

output(3,buffer);

state=0;

i--;

col--;

continue;

}

if(state==14)

{

output(2,buffer);

state=0;

i--;

col--;

continue;

}

}

}

printf("标识符:\n");

for(set<string>::iterator it=S[0].begin(); it!=S[0].end(); it++)

cout<<(\*it)<<' ';

cout<<endl;

printf("无符号数：\n");

for(set<string>::iterator it=S[1].begin(); it!=S[1].end(); it++)

cout<<(\*it)<<' ';

cout<<endl;

printf("关系运算符:\n");

for(set<string>::iterator it=S[2].begin(); it!=S[2].end(); it++)

cout<<(\*it)<<' ';

cout<<endl;

printf("赋值号:\n");

for(set<string>::iterator it=S[5].begin(); it!=S[5].end(); it++)

cout<<(\*it)<<' ';

cout<<endl;

printf("注释：\n");

for(set<string>::iterator it=S[4].begin(); it!=S[4].end(); it++)

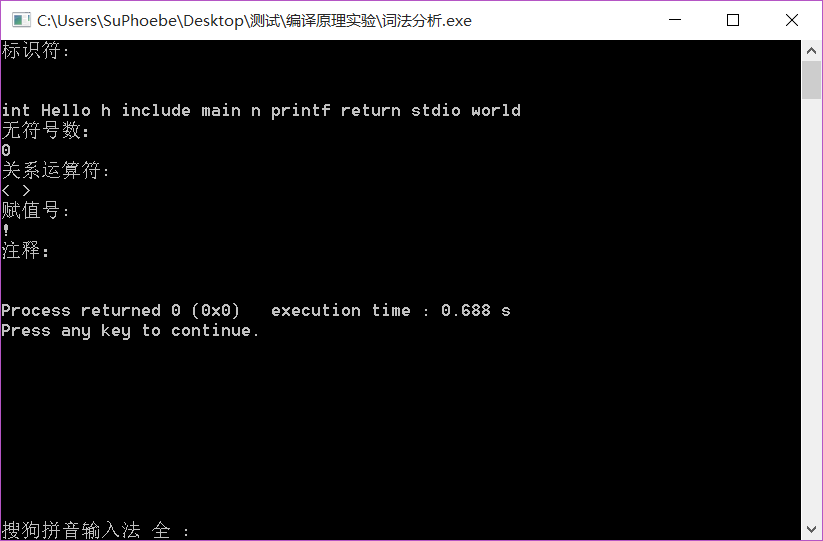
cout<<(\*it)<<' ';

cout<<endl;

return 0;

}

# C++ 代码运行效果截图：

****

**5）lex 代码实现：**

%{

//

// 词法分析.l

// LEX

// Created by 吴浩男 on 15/10/31.

// Copyright © 2015年 吴浩男. All rights reserved.

// Mr.Phoebe的专栏 http://blog.csdn.net/u013007900

#include <stdio.h>

int linenum = 1, wordnum = 0, charnum = 0, colunum = 0, id = 1;

%}

delim [" "|\t]

whitespace {delim}\*

letter [A-Z|a-z|\_]

dight [0-9]

id ({letter})({letter}|{dight})\*

num {dight}\*(\.{dight}\*)?(E[+\-]?{dight}\*)?

comment (\/\\*(.)\*\\*\/)|("//"[^\n]\*)

note \"(\\.|[^\\"])\*\"

%%

{whitespace} {/\*do nothing\*/}

"auto"|"break"|"case"|"char"|"const"|"continue"|"default"|"inc lude"|"do"|"double"|"else"|"enum"|"extern"|"float"|"for"|"goto "|"if"|"int"|"long"|"register"|"return"|"short"|"signed"|"size of"|"static"|"struct"|"switch"|"typedef"|"union"|"unsigned"|"v oid"|"volatile"|"while" {printf("(%s\t C\_UDID)\n", yytext);charnum+=yyleng;wordnum++;}

{id} {printf("(%s\t %d)\n", yytext,id);charnum+=yyleng;wordnum++;id++;}

{num} {printf("(%s\t num)\n",yytext);charnum+=yyleng;}

{comment} {printf("comment: %s\n",yytext);charnum+=yyleng;}

{note} {printf("note %s\n",yytext);charnum+=yyleng;}

">>="|"<<="|"+="|"-="|"\*="|"/="|"%="|"&="|"^="|"|="|">>"|"<<"|"++"|"--"|"&&"|"||"|"<="|">="|"=="|"!="|";"|"{"|"}" |":"|"="|"("|")"|"["|"]"|"."|"&"|"!"|"~"|"-"|"+"|"\*"|"/"|"%"|"<"|">"|"^"|"|"|"?"|","|"#"{printf("(%s sign\n)",yytext);}

[\n] {printf("(newline\t ---)");linenum++;colunum=charnum;charnum++;}

[\r] {printf("(return\t ---)\n");charnum++;}

%%

int yywrap()

{

return 1;

}

int main()

{

if(!(yyin=fopen("data.txt","r")))

{

printf("No such file");

exit(0);

}

yylex();

printf("linenum = %d.\nwordnum = %d.\ncharnum = %d.\n",linenum,wordnum,charnum);

fclose(yyin);

return 0;

}

# lex运行结果截图：

# C:\Users\SuPhoebe\Desktop\测试\编译原理实验\QQ图片20151031204735.png

## **实验心得：**

## 1.这次实验让我对词法分析器有了一定的了解，把理论知识应用于实验中。并且 在实验中灵活运用以前学习过的相关知识进行操作。

## 2．实现功能有限，在注释的实现方面还有待完善，但是我感觉无伤大雅，因为 注释是不会传达给语法分析程序的，个人愚见画蛇添足。

## 3．通过此次编译原理词法分析器的实验，真心激发了我对学习编译原与 Lex 的 积极性，培养了独立发现问题、思考问题，解决问题的能力。通过与老师和同学 的探讨更深刻的掌握了实验原理，了解了至关重要的一步到底是如何运作的。 4．体会到了编译原理与其他学科知识之间的融合渗透。将以前学的 C++ 的知识应用于此次实验，更锻炼了对当前最流行的语言C++编程的操作与想法的强化。