**电子科技大学 信息与软件工程 学院**

**标 准 实 验 报 告**

**（实验）课程名称 编译技术**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名： 程阳 学 号： 2012223010013**

**指导教师： 周尔强**

**实验地点： 信软楼306 实验时间：2013.10.19**

**一、实验室名称：通用计算机软件实验室**

**二、实验项目名称：词法分析器**

**三、实验学时：4学时**

**四、实验原理：**

单词



单词表示的

源程序

词法分析

程序

字符串表示的

源程序

词法分析程序的实现方案

**五、实验目的：**

理解词法分析在编译程序中的作用，掌握词法分析的实现方法和技术，以加深对编译技术中词法分析的理解。

**六、实验内容：**

用C语言（或C++）对一个简单语言编制一个一遍扫描编译程序，扫描源程序字符，按语言的词法规则识别各类单词（如：语言的保留字、标识符、常数）、符号（如运算符、赋值符等），并将有关字符组合成为单词，完成相关转化和处理，最终以二元式形式输出。其中标识符和无符号整数各作为一类，保留字和符号一符一类。具体实验内容包括：

1. 从源程序的第一个字符开始，顺序地读字符，根据所读进的字符识别各类单词；
2. 将所识别的单词分类别分别存储到符号表；其中对常数完成数字字符串到数值的转换；
3. 删去空格、换行、制表等字符和注释。
4. 按要求输出源程序的“单词符号流”，并输出相应符号表的内容，以便检查。

实验输出格式要求

1. 首先输出 自己姓名、学号；
2. 之后输出词法分析结果： 行号 + 词法分析结果二元式， 即：

line xx: (单词类别，单词属性)

1. 如果类别是标识符，还需要输出该标识符在符号表中的内容；
2. 如果类别是常数，还需要输出该常数的值 ；
3. 最后将词法分析结果打印到屏幕并输出到当前目录的 lex.txt 文件
4. 将符号表内容 打印到屏幕并输出到当前目录的 symtab1.txt 文件
5. 将词法错误 打印到屏幕 并 输出到error.txt

**七、实验器材（设备、元器件）：**

硬件要求：pc机，CPU PII 以上，64M 内存，100M 硬盘空间即可。

软件要求：Windows 7/XP/2003等，包括C编译器的IDE。

**八、实验步骤、实验编程与运行结果：**

源码：

**#include** <stdio.h>

**#include** <string.h>

**#include** <stdlib.h>

**#include** <setjmp.h>

**#include** <ctype.h>

**#define** MAX\_ID\_LENGTH **256**

**#define** MAX\_TABLE\_SIZE **2048**

**#define** SUCCESS **1**

**typedef enum** token

**{**

T\_0**=0,**

T\_1**=1,**

T\_2**=2,**

T\_INT**=3,**

T\_IF**=4,**

T\_ELSE**=5,**

T\_WHILE**=6,**

T\_FOR**=7,**

T\_READ**=8,**

T\_WRITE**=9,**

T\_BEGIN**=10,**

T\_FOUNCTION**=11,**

T\_END**=12,**

T\_THEN**=13,**

T\_ADD**=14,**

T\_SUB**=15,**

T\_MUL**=16,**

T\_DIV**=17,**

T\_L**=18,**

T\_LE**=19,**

T\_G**=20,**

T\_GE**=21,**

T\_NE**=22,**

T\_E**=23,**

T\_ASSIGN**=24,**

T\_LPAR**=25,**

T\_RPSR**=25,**

T\_COM**=27,**

T\_SEM**=28,**

T\_RETURN**=29,**

T\_ERROR**=30,**

T\_SYMBOL**=31,**

T\_CONSTANT**=32**

**}** Token**;**

**typedef struct** symtableitem

**{**

**char** name**[**MAX\_ID\_LENGTH**];**

TokenlexToken**;**

**int** type**;**

**bool** init**;**

**}** SymTableItem**;**

**int** line**=1;**

SymTableItemsymtable**[**MAX\_TABLE\_SIZE**];**

**char \***reserved\_table**[] =**

**{**

"integer"**,**

"if"**,**

"else"**,**

"while"**,**

"for"**,**

"read"**,**

"write"**,**

"begin"**,**

"function"**,**

"end"**,**

"then"

**};**

**char\*** operator\_table**[]=**

**{**

"+"**,**

"-"**,**

"\*"**,**

"/"**,**

"<"**,**

"<="**,**

">"**,**

">="**,**

"!="**,**

"=="**,**

":="**,**

"("**,**

")"**,**

","**,**

";"

**};**

**void** InitScanner**();**

**int** ScanOneToken**(**FILE**\*** fp**,char\*** temp**);**

**void** deal\_one\_token**(char\*** temp**);**

**void** deal\_error**(char\*** temp**);**

**void** write\_to\_lex**(char\***temp**,int** count**);**

**void** write\_to\_error**(char\***temp**);**

**void** write\_to\_symtab1**(char\***temp**,int** count**);**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Method: InitScanner**

**// FullName: InitScanner**

**// Access: public**

**// Returns: void**

**// Qualifier:**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void** InitScanner**()**

**{**

**int** count**=0;**

**for(**count**=0;**count**<**MAX\_TABLE\_SIZE**;**count**++)**

**{**

symtable**[**count**].**lexToken**=**T\_1**;**

memset**(**symtable**[**count**].**name**,0,**MAX\_ID\_LENGTH**);**

symtable**[**count**].**init**=0;**

symtable**[**count**].**type**=0;**

**}**

system**(**"del lex.txt symtable1.txt error.txt"**);**

**return ;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Method: ScanOneToken**

**// FullName: ScanOneToken**

**// Access: public**

**// Returns: int**

**// Qualifier:**

**// Parameter: FILE \* fp**

**// Parameter: char \* temp**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**int** ScanOneToken**(**FILE**\*** fp**,char\*** temp**)**

**{**

**int** temp\_count**=0,**is\_operator**=1;**

**char** ch**;**

jmp\_bufbuf1**;**

ch**=**getc**(**fp**);**

**/\* 判D断?是?空?格?和í回?车μ跳?过y\*/**

**while (**ch**==**' '**||**ch**==**'\t'**||**ch**==**'\n'**)**

**{**

**if(**ch**==**'\n'**)**

line**++;**

ch**=**getc**(**fp**);**

**}**

**/\*设Θ?置?还1原-点?，?用?于?数簓字?和í字?符?混ì合?读á取?\*/**

setjmp**(**buf1**);**

**while(**isalpha**(**ch**))/\*如?果?是?字?符?\*/**

**{**

is\_operator**=0;**

temp**[**temp\_count**++]=**ch**;**

**if (**temp\_count**==**MAX\_ID\_LENGTH**-1)**

**{**

printf**(**"error ....缓o冲?溢?出?\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

ch**=**getc**(**fp**);**

**}**

**while(**isdigit**(**ch**))/\*如?果?是?数簓字?\*/**

**{**

is\_operator**=0;**

temp**[**temp\_count**++]=**ch**;**

**if (**temp\_count**==**MAX\_ID\_LENGTH**-1)**

**{**

printf**(**"error ....缓o冲?溢?出?\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

ch**=**getc**(**fp**);**

**if (**isalpha**(**ch**))**

**{**

longjmp**(**buf1**,1);**

**}**

**}**

**if (!**is\_operator**)**

**{**

ungetc**(**ch**,**fp**);/\*回?退?一?个?字?符?，?因皑?为a已?经-识?别纄出?一?个?\*/**

**return** SUCCESS**;**

**}**

**switch(**ch**)/\*处鋦理え?飞え?字?符?和í数簓字?情é况?\*/**

**{**

**case** '>'**:**

**case** '<'**:**

temp**[**temp\_count**++]=**ch**;**

ch**=**getc**(**fp**);**

**if (**ch**==**'='**)**

**{**

temp**[**temp\_count**++]=**ch**;**

**}**

**else**

ungetc**(**ch**,**fp**);**

**return** SUCCESS**;**

**break;**

**case** '!'**:**

**case** ':'**:**

**case** '='**:**

temp**[**temp\_count**++]=**ch**;**

ch**=**getc**(**fp**);**

**if (**ch**!=**'='**)**

**{**

ungetc**(**ch**,**fp**);**

write\_to\_error**(**temp**);**

**return !**SUCCESS**;**

**}**

**else**

temp**[**temp\_count**++]=**ch**;**

**return** SUCCESS**;**

**break;**

**case** '('**:**

**case** ')'**:**

**case** ','**:**

**case** ';'**:**

**case** '+'**:**

**case** '-'**:**

**case** '\*'**:**

**case** '/'**:**

temp**[**temp\_count**++]=**ch**;**

**return** SUCCESS**;**

**break;**

**default:**

temp**[**temp\_count**++]=**ch**;**

write\_to\_error**(**temp**);**

system**(**"pause"**);**

**break;**

**}**

**return !**SUCCESS**;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Method: deal\_one\_token**

**// FullName: deal\_one\_token**

**// Access: public**

**// Returns: void**

**// Qualifier:**

**// Parameter: char \* temp**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void** deal\_one\_token**(char\*** temp**)**

**{**

Tokentemp\_token**;**

**int** count**=0,**help**=0,**is\_digit**=0,**i**=0;**

**/\*判D断?是?不?是?关?键ü字?，?因皑?为a关?键ü字?是?不?能ü作痢?为a标括?示?符?的?\*/**

**for (**count**=0;**count**<11;**count**++)**

**{**

**if (!**strcmp**(**temp**,**reserved\_table**[**count**]))**

**{**

printf**(**"%s\t\t对?应畖于睺OKEN序ò列：阰 %d\n"**,**temp**,**count**+3);**

write\_to\_lex**(**temp**,**count**+3);**

**return ;**

**}**

**}**

**/\*判D断?是?不?是?运?算?符?\*/**

**for (**count**=0;**count**<15;**count**++)**

**{**

**if (!**strcmp**(**operator\_table**[**count**],**temp**))**

**{**

printf**(**"%s\t 存?在ú于?运?算?符?中D\t对?应畖于睺OKEN序ò列：阰 %d\n"**,**temp**,**count**+10);**

write\_to\_lex**(**temp**,**count**+10);**

**return ;**

**}**

**}**

**for(**count**=0;**symtable**[**count**].**init**&&(**count**<**MAX\_TABLE\_SIZE**);**count**++)**

**{**

**if (!**strcmp**(**symtable**[**count**].**name**,**temp**))**

**{**

printf**(**"%s\t存?在ú于?标括?示?符?的?位?置?: %d\t对?应畖的?TOKEN序ò列是?：阰%d\n"**,**temp**,**count**,**symtable**[**count**].**lexToken**);**

write\_to\_lex**(**temp**,**symtable**[**count**].**lexToken**);**

**return ;**

**}**

**}**

**if (**count**==**MAX\_TABLE\_SIZE**)**

**{**

printf**(**"缓o冲?区?域?溢?出?\n"**);**

exit**(**EXIT\_FAILURE**);**

system**(**"pause"**);**

**return;**

**}**

write\_to\_symtab1**(**temp**,**count**);**

printf**(**"%s\t对?应畖的?TOKEN序ò列是?：阰%d\n"**,**temp**,**symtable**[**count**].**lexToken**);**

write\_to\_lex**(**temp**,**symtable**[**count**].**lexToken**);**

**return ;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Method: write\_to\_error**

**// FullName: write\_to\_error**

**// Access: public**

**// Returns: void**

**// Qualifier:**

**// Parameter: char \* temp**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void** write\_to\_error**(char\***temp**)**

**{**

FILE **\***ERROR\_FILE**=**fopen**(**"error.txt"**,**"a+"**);**

**if (!**ERROR\_FILE**)**

**{**

printf**(**"can`t open file error.txt exit\_failure\n"**);**

system**(**"pause"**);**

exit**(**EXIT\_FAILURE**);**

**}**

fprintf**(**ERROR\_FILE**,**"行D%d\t未′识?别纄字?符?组哩?合?%s\n"**,**line**,**temp**);**

fclose**(**ERROR\_FILE**);**

**return;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Method: write\_to\_lex**

**// FullName: write\_to\_lex**

**// Access: public**

**// Returns: void**

**// Qualifier:**

**// Parameter: char \* temp**

**// Parameter: int count**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void** write\_to\_lex**(char\***temp**,int** count**)**

**{**

FILE**\*** LEX\_FILE**=**fopen**(**"lex.txt"**,**"a+"**);**

**if (!**LEX\_FILE**)**

**{**

printf**(**"error in open file lex.txt exit failure\n"**);**

system**(**"pause"**);**

**}**

fprintf**(**LEX\_FILE**,**"行D%d\t字?符?：阰%s\t对?应畖TOKEN序ò列：阰%d\n"**,**line**,**temp**,**count**);**

fclose**(**LEX\_FILE**);**

**return;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Method: write\_to\_symtab1**

**// FullName: write\_to\_symtab1**

**// Access: public**

**// Returns: void**

**// Qualifier:**

**// Parameter: char \* temp**

**// Parameter: int count**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void** write\_to\_symtab1**(char\***temp**,int** count**)**

**{**

**int** help**=0,**is\_digit**=0;**

**for (**help**=0;**help**<**MAX\_ID\_LENGTH**;**help**++)**

**{**

**if (**temp**[**help**]==**'\0'**)**

**{**

**break;**

**}**

**if (**isdigit**(**temp**[**help**]))**

**{**

is\_digit**=1;**

**}**

**else**

**{**

is\_digit**=0;**

**break;**

**}**

**}**

strcat**(**symtable**[**count**].**name**,**temp**);**

symtable**[**count**].**init**=1;**

**if (**is\_digit**)**

**{**

symtable**[**count**].**lexToken**=**T\_CONSTANT**;**

symtable**[**count**].**type**=**atoi**(**temp**);**

**}**

**else**

**{**

symtable**[**count**].**lexToken**=**T\_SYMBOL**;**

**}**

**return;**

**}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// Method: main**

**// FullName: main**

**// Access: public**

**// Returns: int**

**// Qualifier: 主÷函ˉ数簓**

**// Parameter: int agrc**

**// Parameter: char \* argv[]**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**int** main**(int** agrc**,char \***argv**[])**

**{**

printf**(**"\t\t2012223010013 程ì阳?\n"**);**

FILE**\*** symtable1**=**NULL**;**

**int** num**=0;**

**if(**agrc**<2)**

**{**

printf**(**"ERROR: source file needed.\nUSAGE: mylex source.c\n"**);**

**return 0;**

**}**

FILE**\*** fp**=**fopen**(**argv**[1],**"r"**);**

FILE**\*** file\_read**=**fp**;**

**// char source\_from[20]={0};**

**char** temp**[**MAX\_ID\_LENGTH**];**

**// strcat(source\_from,"e:\\test.txt");**

**// file\_read=fopen(source\_from,"r");**

**if(!**file\_read**)**

**{**

printf**(**"error in open source\n"**);**

exit**(0);**

**}**

InitScanner**();**

**for (;;)**

**{**

**if (**feof**(**file\_read**))**

**{**

**break;**

**}**

memset**(**temp**,0,sizeof(**temp**));**

ScanOneToken**(**file\_read**,**temp**);**

printf**(**"line= %d\n"**,**line**);**

deal\_one\_token**(**temp**);**

**}**

**if (!(**symtable1**=**fopen**(**"symtable1.txt"**,**"a+"**)))**

**{**

printf**(**"can`t open file symtable1 exit failure\n"**);**

system**(**"pause"**);**

**}**

**for (**num**=0;(**num**<**MAX\_TABLE\_SIZE**)&&(**symtable**[**num**].**init**);**num**++)**

**{**

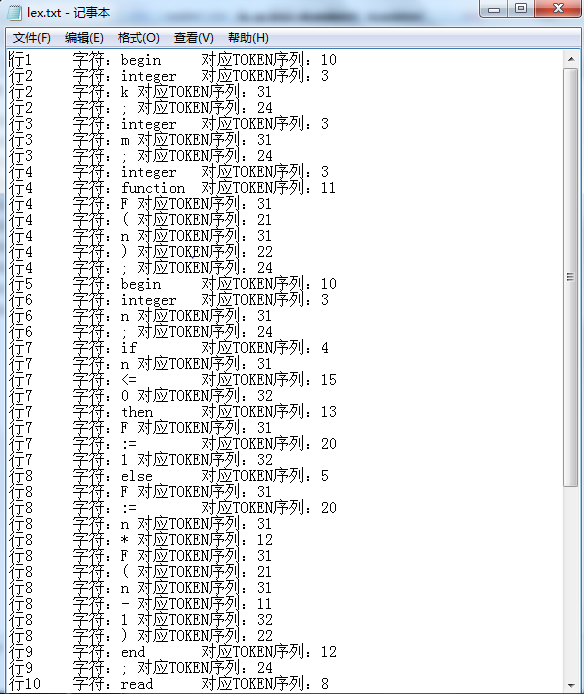
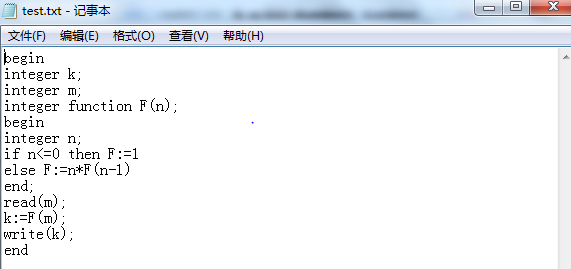
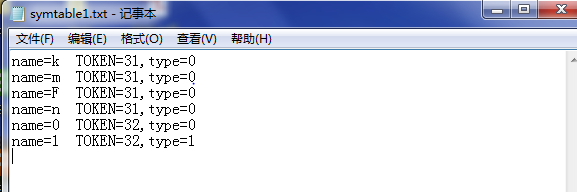
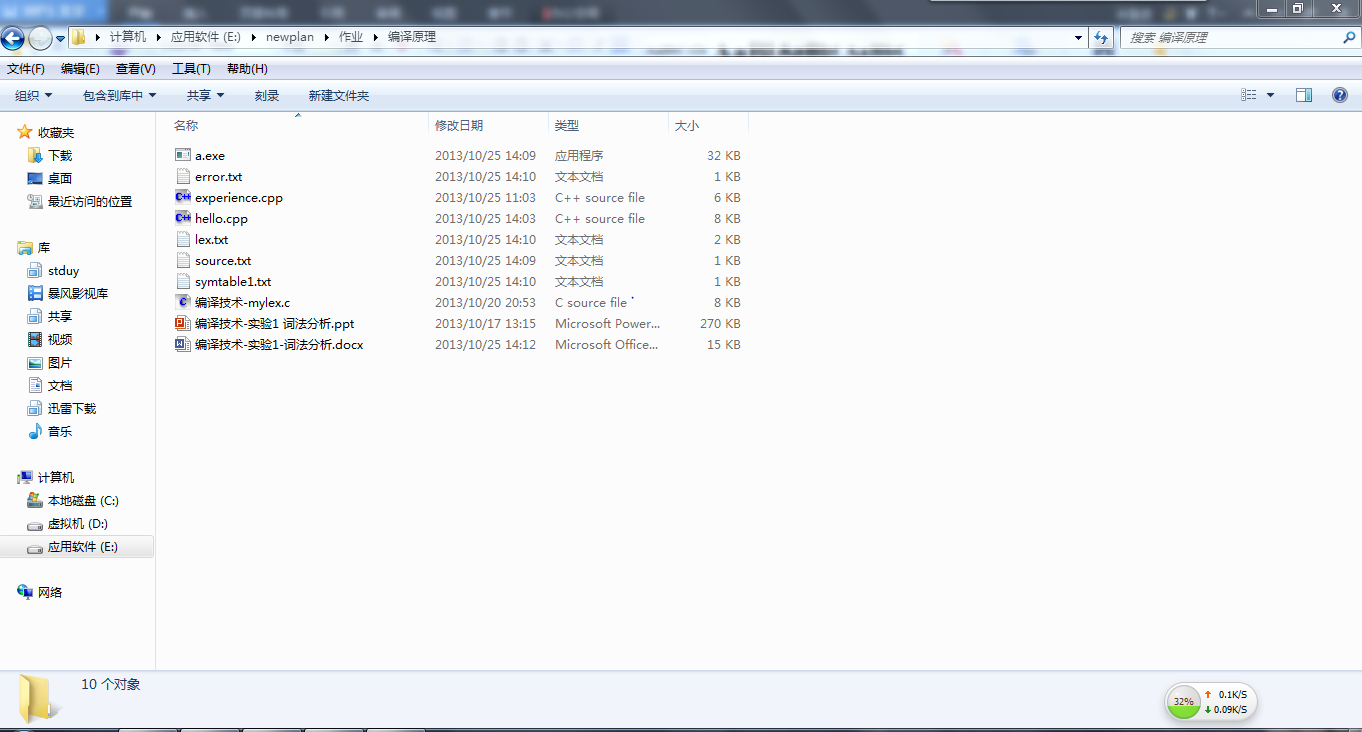
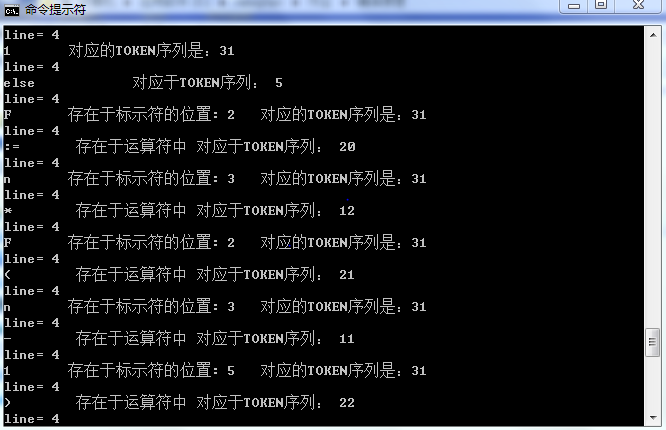
fprintf**(**symtable1**,**"name=%s\tTOKEN=%d,type=%d\n"**,**symtable**[**num**].**name**,**symtable**[**num**].**lexToken**,**symtable**[**num**].**type**);**

**}**

fclose**(**file\_read**);**

**return 0;**

**}**



**九、实验结论：**

用C语言（或C++）对一个简单语言编制一个一遍扫描编译程序，扫描源程序字符，按语言的词法规则识别各类单词（如：语言的保留字、标识符、常数）、符号（如运算符、赋值符等），并将有关字符组合成为单词，完成相关转化和处理，最终以二元式形式输出。其中标识符和无符号整数各作为一类，保留字和符号一符一类。具体实验内容包括：

1从源程序的第一个字符开始，顺序地读字符，根据所读进的字符识别各类单词；

2将所识别的单词分类别分别存储到符号表；其中对常数完成数字字符串到数值的转换；

3删去空格、换行、制表等字符。

4按要求输出源程序的“单词符号流”，并输出相应符号表的内容，以便检查。实验输出格式要求

5首先输出 自己姓名、学号；

6之后输出词法分析结果： 行号 + 词法分析结果二元式， 即：line xx: (单词类别，单词属性)

7如果类别是标识符，还需要输出该标识符在符号表中的内容；

8如果类别是常数，还需要输出该常数的值 ；

9最后将词法分析结果打印到屏幕并输出到当前目录的 lex.txt 文件

10将符号表内容 打印到屏幕并输出到当前目录的 symtab1.txt 文件

11将词法错误 打印到屏幕 并 输出到error.txt

**十、总结及心得体会：**

本实验的练习使我明白了如何去构建词法分析器，以及文字的编码格式问题，在今后的编程练习更加要注意编码问题。

同时，也对C语言的知识复习一遍，让我懂得基础的重要。

本实验还锻炼了我们独立解决问题的方法和能力

**十一、对本实验过程及方法、手段的改进建议：**

本实验中需要注意的事项还有很多，因为根据编码的不同，导致了许多问题的出现，同时，在一些算法设计上还有很大一部分的不足，本程序用于较小的地方还是可以接受的，但是要编译大量文件，程序的健壮性和高效性还面临很大的挑战！

**报告评分：**

**指导教师签字：**