

编译原理

**词法分析器实验报告**

**Lexical Analyzer Experimental Report**

学 院： 计算机与信息技术

专 业： 计算机科学

学生姓名： 刘宜进

学 号： 14282008

指导教师： 徐金安

**北京交通大学**

2017年5月

目 录

[目 录 ii](#_Toc483338074)

[1 实验目的 3](#_Toc483338075)

[2 实验内容 3](#_Toc483338076)

[2.1程序功能描述 3](#_Toc483338077)

[2.2程序结构 3](#_Toc483338078)

[2.2.1 源程序读取 4](#_Toc483338079)

[2.2.2 词法分析 4](#_Toc483338080)

[2.2.3 界面展示 4](#_Toc483338081)

[2.3数据结构 4](#_Toc483338082)

[2.4主要函数 5](#_Toc483338083)

[3 程序测试 5](#_Toc483338084)

[3.1测试用例 5](#_Toc483338085)

[3.2测试结果 6](#_Toc483338086)

[3.3 结果分析 6](#_Toc483338087)

[附 录 7](#_Toc483338088)

1. 实验目的

以下为正则文法所描述的 C 语言子集单词符号的示例，请补充单词符号：

++，--， >>, <<, += , -= ,\*=, /= ，&&（逻辑与），||（逻辑或），！（逻辑非）等等，给出补 充后描述 C 语言子集单词符号的正则文法，设计并实现其词法分析程序。

1. 给出各单词符号的类别编码；
2. 词法分析程序应能发现输入串中的错误；
3. 词法分析作为单独一遍编写，词法分析结果为二元式序列组成的中间文件；
4. 设计两个测试用例（尽可能完备），并给出测试结果。
5. 实验内容

该实验运用词法分析基本原理，利用Python语言实现了一个C程序的词法分析器。下面，通过程序功能描述、程序结构、数据结构、主要函数等四方面展开详细介绍。

2.1程序功能描述

该程序能够从本地读取C程序，并进行简单的词法分析，并以四元组的形式输出。同时，对于程序中的一些词法错误，能够进行识别。该程序最终生成图形用户接口(GUI)，方便操作。

2.2程序结构

该程序主要有三大部分组成：

1. 源程序读取
2. 词法分析
3. 界面展示

2.2.1 源程序读取

源程序的读取采用Python文件读取流。

root**.**filename **=** filedialog**.**askopenfilename**(**

initialdir **=** "/"**,**

title **=** "Select file"**,**

filetypes **=** **((**"c files"**,**"\*.c"**),**

**(**"all files"**,**"\*.\*"**)))**

fin **=** open**(**root**.**filename**,** "r"**)**

接着以换行符为分割符，将源程序的每一行进行分割，并存贮在一个列表List中。

同时，提供一个读取每一个字符的函数getchar(input\_str)。

2.2.2 词法分析

词法分析又包括了扫描程序、错误报告、四元组生成三部分

扫描程序，每次读取一个字符，判断它是标识符、数字、分隔符或者操作符。

错误报告，每当检测到错误，将错误信息打印。

四元组生成，将分析正确的结果以四元组的形式生成。

2.2.3 界面展示

主要利用Python自带库tkinter，主体由四部分组成：

CodeTest:该组件是可编辑文本，对源程序进行展示；

ErrorTest:该组件也是文本，对检测出的错误进行展示；

Analysis：该组件为分析的结果，即四元组。

Menu:该组件是菜单，由四个子菜单键filemenu、lexmenu、windowsmenu和helpmenu。

Filemenu是文件子菜单，包括文件打开、保存、退出等；

Lexmenu 是分析子菜单，只有一个下拉选项lex，点击进行词法分析；

Windowsmenu 是窗口子菜单，点击fullscreen可全屏，或者按快捷键F11；

Helpmenu 是帮助子菜单，目前没有实现。

2.3数据结构

首先用三个列表(List)分别表示关键字、操作符、分隔符

用一个字典(Dict)表示字符表及其编码

两个全局变量current\_row、current\_line表示当前行和列，用来取字符和定位错误

2.4主要函数

表2-1 主要函数及功能介绍

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 参数 | 返回值 | 用途 |
| getchar | input\_str：string | 下一个字符 | 读取一个字符 |
| ungetchar | input\_str：string | 上一个字符 | 退回一个字符 |
| error | msg：strign | no | 错误 |
| scanner | input\_str：string | no | 词法分析 |
| fileloader | no | no | 打开源程序 |
| pre\_interface | no | no | 前端界面 |

1. 程序测试

3.1测试用例

采用两个C代码分别进行测试，第一是冒泡排序，其中存在错误如下：

float f **=** 2e10a**;**

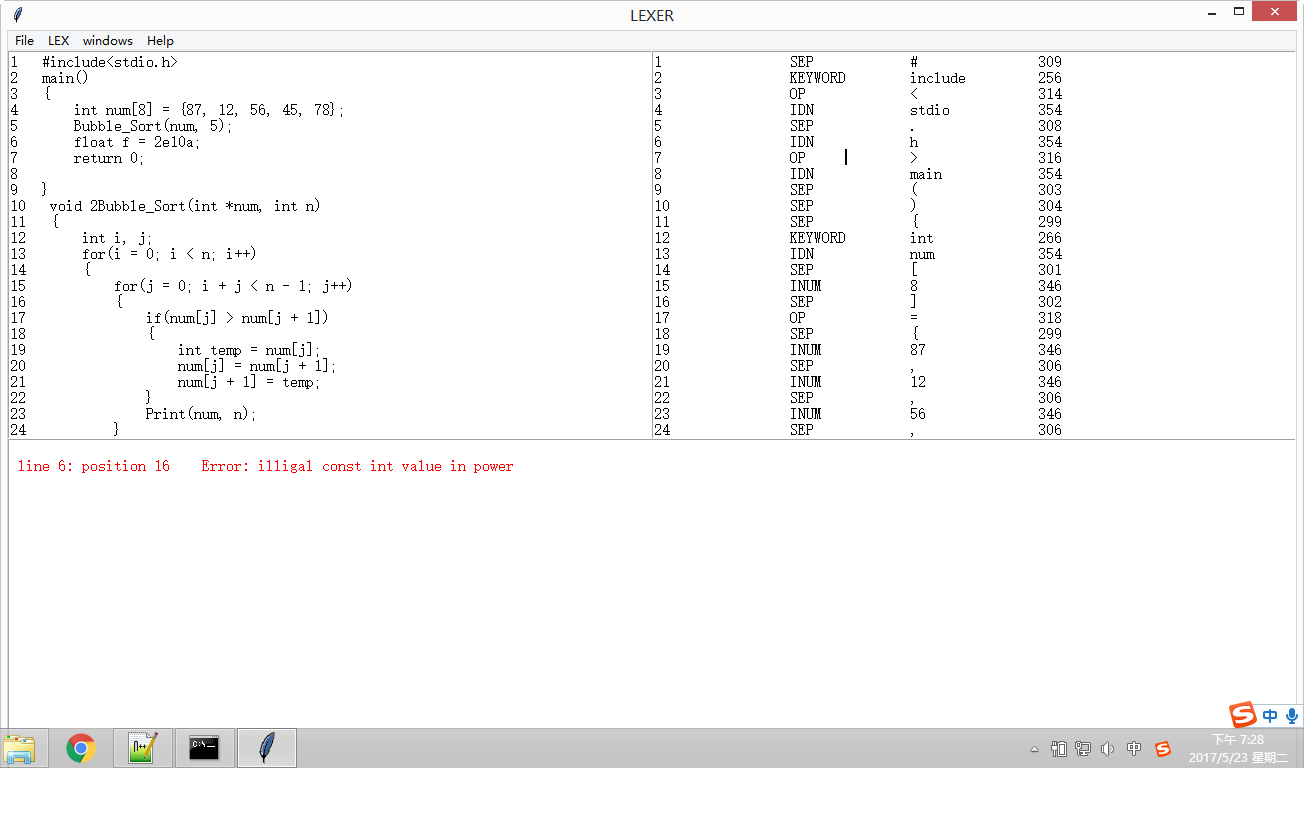
void 2Bubble\_Sort**(**int **\***num**,** int n**)**

第二个是杨辉三角，其中存在错误如下：

3main**()**

分别进行测试，并对测试结果进行测评。

3.2测试结果



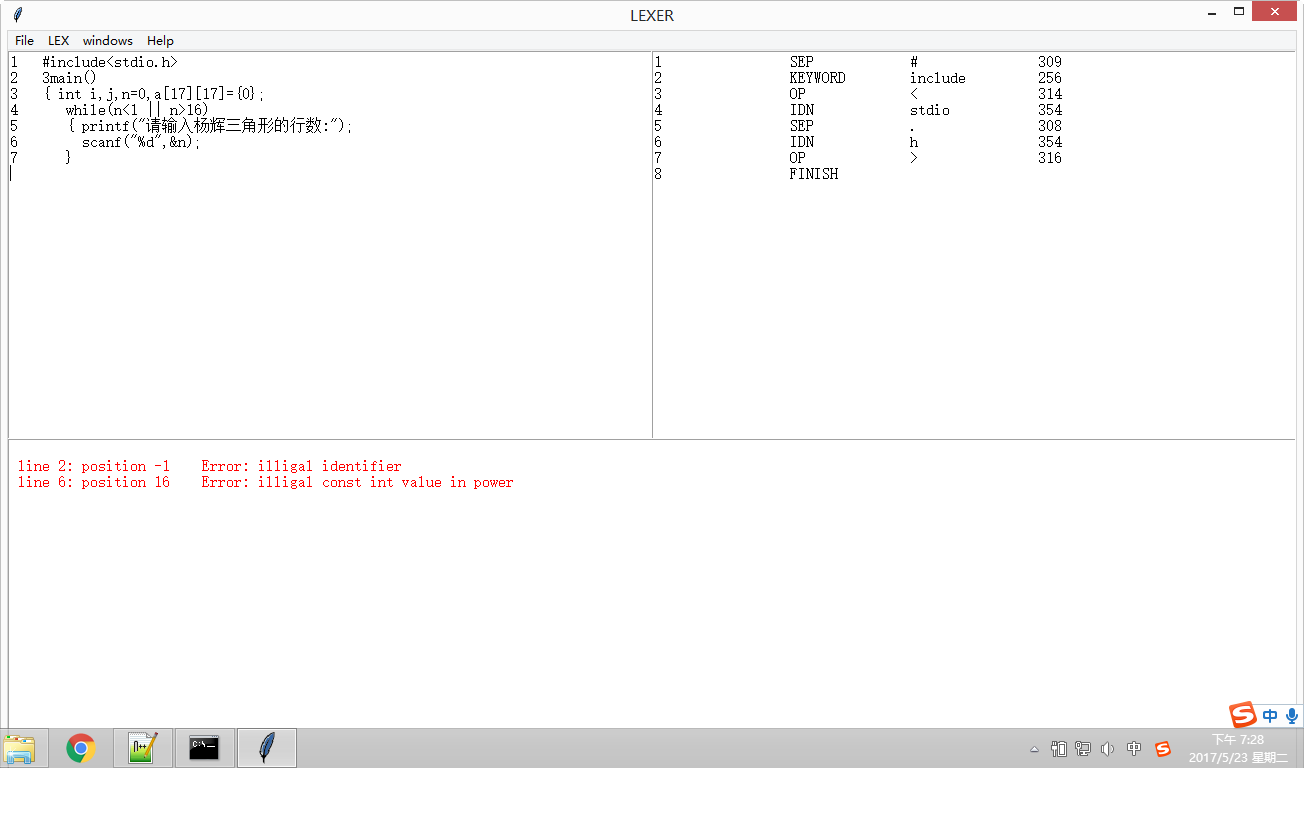
图3-1 冒泡排序结果

图3-2 杨辉三角结果

3.3 结果分析

采用两个C代码分别进行测试，第一是冒泡排序，其中存在错误如下：

根据上图可以看出，程序正确的识别出了源程序中的词法错误，并提示出来。同时，对于分析结果四元组的形式进行展示。

附 录

附录A 程序代码

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

**import** sys

**from** tkinter **import** filedialog

**from** tkinter **import** **\***

KEYWORD\_LIST **=** **[**'if'**,** 'else'**,** 'while'**,** 'break'**,** 'continue'**,** 'for'**,** 'double'**,** 'int'**,** 'float'**,** 'long'**,** 'short'**,** 'bool'**,**'switch'**,** 'case'**,** 'return'**,** 'void'**,**'include'**]**

SEPARATOR\_LIST **=** **[**'{'**,** '}'**,** '['**,** ']'**,** '('**,** ')'**,** '~'**,** ','**,** ';'**,** '.'**,** '?'**,** ':'**,** ' '**,**'#'**]**

OPERATOR\_LIST **=** **[**'+'**,** '++'**,** '-'**,** '--'**,** '+='**,** '-='**,** '\*'**,** '\*='**,** '%'**,** '%='**,** '->'**,** '|'**,** '||'**,** '|='**,**'/'**,** '/='**,** '>'**,** '<'**,** '>='**,** '<='**,** '='**,** '=='**,** '!='**,** '!'**,** '&'**,** '\*\*'**]**

CATEGORY\_DICT **=** **{**

# KEYWORD

"include"**:**256**,**

"far"**:** 257**,**

"near"**:** 258**,**

"pascal"**:** 259**,**

"register"**:** 260**,**

"asm"**:** 261**,**

"cdecl"**:** 262**,**

"huge"**:** 263**,**

"auto"**:** 264**,**

"double"**:** 265**,**

"int"**:** 266**,**

"struct"**:** 267**,**

"break"**:** 268**,**

"else"**:** 269**,**

"long"**:** 270**,**

"switch"**:** 271**,**

"case"**:** 272**,**

"enum"**:** 273**,**

"register"**:** 274**,**

"typedef"**:** 275**,**

"char"**:** 276**,**

"extern"**:** 277**,**

"return"**:** 278**,**

"union"**:** 279**,**

"const"**:** 280**,**

"float"**:** 281**,**

"short"**:** 282**,**

"unsigned"**:** 283**,**

"continue"**:** 284**,**

"for"**:** 285**,**

"signed"**:** 286**,**

"void"**:** 287**,**

"default"**:** 288**,**

"goto"**:** 289**,**

"sizeof"**:** 290**,**

"volatile"**:** 291**,**

"do"**:** 292**,**

"if"**:** 293**,**

"while"**:** 294**,**

"static"**:** 295**,**

"interrupt"**:** 296**,**

"sizeof"**:** 297**,**

"NULL"**:** 298**,**

# SEPARATOR

"{"**:** 299**,**

"}"**:** 300**,**

"["**:** 301**,**

"]"**:** 302**,**

"("**:** 303**,**

")"**:** 304**,**

"~"**:** 305**,**

","**:** 306**,**

";"**:** 307**,**

"."**:** 308**,**

"#"**:** 309**,**

"?"**:** 310**,**

":"**:** 311**,**

# OPERATOR

"<<"**:** 312**,**

">>"**:** 313**,**

"<"**:** 314**,**

"<="**:** 315**,**

">"**:** 316**,**

">="**:** 317**,**

"="**:** 318**,**

"=="**:** 319**,**

"|"**:** 320**,**

"||"**:** 321**,**

"|="**:** 322**,**

"^"**:** 323**,**

"^="**:** 324**,**

"&"**:** 325**,**

"&&"**:** 326**,**

"&="**:** 327**,**

"%"**:** 328**,**

"%="**:** 329**,**

"+"**:** 330**,**

"++"**:** 331**,**

"+="**:** 332**,**

"-"**:** 333**,**

"--"**:** 334**,**

"-="**:** 335**,**

"->"**:** 336**,**

"/"**:** 337**,**

"/="**:** 338**,**

"\*"**:** 339**,**

"\*="**:** 340**,**

"!"**:** 341**,**

"!="**:** 342**,**

"sizeof"**:** 343**,**

"<<="**:** 344**,**

">>="**:** 345**,**

"inum"**:** 346**,**

"int16"**:** 347**,**

"int8"**:** 348**,**

"char"**:** 350**,**

"string"**:** 351**,**

"bool"**:** 352**,**

"fnum"**:** 353**,**

"IDN"**:** 354**,**

'\*\*'**:** 355

**}**

current\_row **=** **-**1

current\_line **=** 0

out\_line **=** 1

**global** errorTest

# 读取一个字符

**def** getchar**(**input\_str**):**

**global** current\_row

**global** current\_line

current\_row **+=** 1

**if** **(**current\_row **==** len**(**input\_str**[**current\_line**])):**

current\_line **+=** 1

current\_row **=** 0

**if** current\_line **==** len**(**input\_str**):**

**return** 'FINISH'

**return** input\_str**[**current\_line**][**current\_row**]**

# 退格

**def** ungetchar**(**input\_str**):**

**global** current\_row

**global** current\_line

current\_row **=** current\_row **-** 1

**if** current\_row **<** 0**:**

current\_line **=** current\_line **-** 1

current\_row **=** len**(**input\_str**[**current\_row**])** **-** 1

**return** input\_str**[**current\_line**][**current\_row**]**

# 错误报告

**def** error**(**msg**,** line**=None,** row**=None):**

**global** out\_line

**global** errorTest

**if** line **is** **None:**

line **=** current\_line **+** 1

**if** row **is** **None:**

row **=** current\_row **+** 1

errorTest**.**insert**(**str**(**out\_line**)** **+** '.end'**,** " line "**+** str**(**line**)** **+** ': position ' **+** str**(**row**-**4**)** **+** '\t\t\tError: ' **+** msg**)**

errorTest**.**insert**(**str**(**out\_line**)** **+** '.end'**,** "\n"**)**

out\_line **=** out\_line **+** 1

# 扫描器

**def** scanner**(**input\_str**):**

**global** current\_line

**global** current\_row

current\_char **=** getchar**(**input\_str**)**

**if** current\_char **==** 'FINISH'**:**

**return** **(**'FINISH'**,** ''**,** ''**)**

**if** current\_char**.**strip**()** **==** ''**:**

**return**

# 数字

**if** current\_char**.**isdigit**():**

int\_value **=** 0

**while** current\_char**.**isdigit**():**

int\_value **=** int\_value **\*** 10 **+** int**(**current\_char**)**

current\_char **=** getchar**(**input\_str**)**

**if** current\_char **not** **in** OPERATOR\_LIST **and** current\_char **not** **in** SEPARATOR\_LIST **and** current\_char **!=** 'e'**:**

line **=** current\_line **+** 1

row **=** current\_row **+** 1

ungetchar**(**input\_str**)**

error**(**'illigal identifier'**,** line**,** row**)**

**return** **(**'FINISH'**,** ''**,** ''**)**

**return** **None**

**if** current\_char **!=** '.' **and** current\_char **!=** 'e'**:**

ungetchar**(**input\_str**)**

**return** **(**'INUM'**,** int\_value**,** CATEGORY\_DICT**[**'inum'**])**

**if** current\_char **==** 'e'**:**

power\_value **=** str**(**int\_value**)** **+** 'e'

current\_char **=** getchar**(**input\_str**)**

**if** current\_char **==** '+' **or** current\_char **==** '-'**:**

power\_value **+=** current\_char

current\_char **=** getchar**(**input\_str**)**

**while** current\_char**.**isdigit**():**

power\_value **+=** current\_char

current\_char **=** getchar**(**input\_str**)**

**if** current\_char **not** **in** OPERATOR\_LIST **and** current\_char **not** **in** SEPARATOR\_LIST**:**

line **=** current\_line **+** 1

row **=** current\_row **+** 1

ungetchar**(**input\_str**)**

error**(**'illigal const int value in power'**,** line**,** row**)**

**return** **(**'FINISH'**,** ''**,** ''**)**

**return** **None**

ungetchar**(**input\_str**)**

**return** **(**'INUM'**,** power\_value**,** CATEGORY\_DICT**[**'inum'**])**

**if** current\_char **==** '.'**:**

float\_value **=** str**(**int\_value**)** **+** '.'

current\_char **=** getchar**(**input\_str**)**

**while** current\_char**.**isdigit**():**

float\_value **+=** current\_char

current\_char **=** getchar**(**input\_str**)**

**if** current\_char **not** **in** OPERATOR\_LIST **and** current\_char **not** **in** SEPARATOR\_LIST **or** current\_char **==** '.'**:**

line **=** current\_line **+** 1

row **=** current\_row **+** 1

ungetchar**(**input\_str**)**

error**(**'illigal const float value'**,** line**,** row**)**

**return** **(**'FINISH'**,** ''**,** ''**)**

**return** **None**

ungetchar**(**input\_str**)**

**return** **(**'FNUM'**,** float\_value**,** CATEGORY\_DICT**[**'fnum'**])**

# 标识符

**if** current\_char**.**isalpha**()** **or** current\_char **==** '\_'**:**

string **=** ''

**while** current\_char**.**isalpha**()** **or** current\_char**.**isdigit**()** **or** current\_char **==** '\_' **and** current\_char **!=** ' '**:**

string **+=** current\_char

current\_char **=** getchar**(**input\_str**)**

**if** current\_char **==** 'FINISH'**:**

**break**

ungetchar**(**input\_str**)**

**if** string **in** KEYWORD\_LIST**:**

**return** **(**"KEYWORD"**,** string**,** CATEGORY\_DICT**[**string**])**

**else:**

**return** **(**'IDN'**,** string**,** CATEGORY\_DICT**[**'IDN'**])**

# 注释

**if** current\_char **==** '\"'**:**

str\_literal **=** ''

line **=** current\_line **+** 1

row **=** current\_row **+** 1

current\_char **=** getchar**(**input\_str**)**

**while** current\_char **!=** '\"'**:**

str\_literal **+=** current\_char

current\_char **=** getchar**(**input\_str**)**

**if** current\_char **==** 'FINISH'**:**

error**(**'missing terminating \"'**,** line**,** row**)**

current\_line **=** line

current\_row **=** row

**return** **(**'FINISH'**,** ''**,** ''**)**

**return** **(**'STRING\_LITERAL'**,** str\_literal**,** CATEGORY\_DICT**[**'string'**])**

**if** current\_char **==** '/'**:**

next\_char **=** getchar**(**input\_str**)**

line **=** int**(**current\_line**)** **+** 1

row **=** int**(**current\_row**)** **+** 1

**if** next\_char **==** '\*'**:**

comment **=** ''

next\_char **=** getchar**(**input\_str**)**

**while** **True:**

**if** next\_char **==** 'FINISH'**:**

error**(**'unteminated /\* comment'**,** line**,** row**)**

**return** **(**'FINISH'**,** ''**,** ''**)**

**if** next\_char **==** '\*'**:**

end\_char **=** getchar**(**input\_str**)**

**if** end\_char **==** '/'**:**

**return** **None**

**if** end\_char **==** 'FINISH'**:**

error**(**'unteminated /\* comment'**,** line**,** row**)**

**return** **(**'FINISH'**,** ''**,** ''**)**

comment **+=** next\_char

next\_char **=** getchar**(**input\_str**)**

**else:** #/=

ungetchar**(**input\_str**)**

op **=** current\_char

current\_char **=** getchar**(**input\_str**)**

**if** current\_char **in** OPERATOR\_LIST**:**

op **+=** current\_char

**else:** # /

ungetchar**(**input\_str**)**

**return** **(**'OP'**,** op**,** CATEGORY\_DICT**[**op**])**

**if** current\_char **in** SEPARATOR\_LIST**:**

**return** **(**'SEP'**,** current\_char**,** CATEGORY\_DICT**[**current\_char**])**

**if** current\_char **in** OPERATOR\_LIST**:**

op **=** current\_char

current\_char **=** getchar**(**input\_str**)**

**if(**current\_char **in** OPERATOR\_LIST**):**

op **+=** current\_char

**else:**

ungetchar**(**input\_str**)**

**return** **(**'OP'**,** op**,** CATEGORY\_DICT**[**op**])**

**else:**

error**(**'unknown character: ' **+** current\_char**)**

**def** fileloader**():**

**global** root

code**.**delete**(**1.0**,** END**)**

root**.**filename **=** filedialog**.**askopenfilename**(**

initialdir **=** "/"**,**

title **=** "Select file"**,**

filetypes **=** **((**"c files"**,**"\*.c"**),**

**(**"all files"**,**"\*.\*"**)))**

fin **=** open**(**root**.**filename**,** "r"**)**

input\_file **=** fin**.**read**()**

input\_line **=** input\_file**.**split**(**"\n"**)**

out\_line **=** 1

**for** each **in** input\_line**:**

code**.**insert**(**str**(**out\_line**)** **+** '.end'**,** str**(**out\_line**)+** " " **+** each**)**

code**.**insert**(**str**(**out\_line**)** **+** '.end'**,** "\n"**)**

out\_line **=** out\_line **+** 1

fin**.**close**()**

#　词法分析

**def** lexer\_analysis**(**input\_str**):**

**global** current\_row

**global** current\_line

**global** out\_line

current\_row **=** **-**1

current\_line **=** 0

analysis\_result **=** **[]**

r **=** **[**''**,**''**,**''**]**

**while** **(**1**):**

r **=** scanner**(**input\_str**)**

**if** r **is** **not** **None:**

analysis\_result**.**append**(**str**(**r**[**0**])** **+** "\t\t" **+** str**(**r**[**1**])** **+** "\t\t" **+** str**(**r**[**2**]))**

**if** **(**r**[**0**]** **==** 'FINISH'**):**

**return** analysis\_result

**return** analysis\_result

#　按键触发函数

**def** lexer**():**

**global** out\_line

input\_str **=** **[]**

analysis**.**delete**(**1.0**,** END**)**

input\_raw **=** code**.**get**(**1.0**,** END**)**

input\_str **=** input\_raw**.**split**(**"\n"**)**

temp **=** **[]**

**for** i **in** range**(**len**(**input\_str**)):**

input\_str**[**i**]=**input\_str**[**i**][**3**:]** #remove the line number

**if** **(**input\_str**[**i**]!=** ""**):**

temp**.**append**(**input\_str**[**i**])**

out\_line **=** 0

result **=** lexer\_analysis**(**temp**)**

**for** each **in** result**:**

analysis**.**insert**(**str**(**out\_line**)** **+** '.end'**,** str**(**out\_line**)** **+** " \t\t "**+** each**)**

analysis**.**insert**(**str**(**out\_line**)** **+** '.end'**,** "\n"**)**

out\_line **=** out\_line **+** 1

# 界面展示

**def** pre\_interface**():**

**global** root

**global** code

**global** analysis

**global** errorTest

root **=** Tk**()**

menubar **=** Menu**(**root**)**

filemenu **=** Menu**(**menubar**,** tearoff**=**0**)**

filemenu**.**add\_command**(**label**=**"Open"**,** command**=**fileloader**,**font **=** 26**)**

filemenu**.**add\_command**(**label**=**"Save" **,**font **=** 26**)**

filemenu**.**add\_command**(**label**=**"Exit"**,** command**=**root**.**quit**,**font **=** 26**)**

menubar**.**add\_cascade**(**label**=**"File"**,** menu**=**filemenu**,**font **=** 26**)**

lexmenu **=** Menu**(**menubar**,** tearoff**=**0**)**

lexmenu**.**add\_command**(**label**=**"lex"**,** command**=**lexer**,**font **=** 26**)**

menubar**.**add\_cascade**(**label**=**"LEX"**,** menu**=**lexmenu**,**font **=** 26**,**command **=** root**.**quit**)**

windowsmenu **=** Menu**(**menubar**,** tearoff**=**0**)**

windowsmenu**.**add\_command**(**label**=**"fullscreen"**,** command**=**toggle\_fullscreen**,**font **=** 26**)**

menubar**.**add\_cascade**(**label**=**"windows"**,** menu**=**windowsmenu**,**font **=** 26**)**

helpmenu **=** Menu**(**menubar**,** tearoff**=**0**)**

helpmenu**.**add\_command**(**label**=**"Help Index"**,**font **=** 26**)**

menubar**.**add\_cascade**(**label**=**"Help"**,** menu**=**helpmenu**,**font **=** 26**)**

root**.**config**(**menu**=**menubar**)**

code **=** Text**(**root**,** font**=**26**)**

analysis **=** Text**(**root**,** font**=**26**)**

errorTest **=** Text**(**root**,** width **=** 10**,** font**=**26**,**foreground**=**"red"**)**

root**.**title**(**"LEXER"**)**

errorTest**.**pack**(**fill **=** X**,**side**=**BOTTOM**,**expand **=** YES**)**

code**.**pack**(**side **=**LEFT**,**fill **=** Y**,**expand **=** YES**)**

analysis**.**pack**(**side**=**RIGHT**,**fill **=** Y**,**expand **=** YES**)**

root**.**bind**(**"<F11>"**,** toggle\_fullscreen**)**

root**.**mainloop**()**

**def** toggle\_fullscreen**(**event**=None):** # 增加全屏属性

root**.**state**(**"zoomed"**)**

**def** main**():**

pre\_interface**()**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

main**()**

附录B 测试代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ðŝŅв\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

main**()**

**{**

int num**[**8**]** **=** **{**87**,** 12**,** 56**,** 45**,** 78**};**

Bubble\_Sort**(**num**,** 5**);**

float f **=** 2e10a**;**

**return** 0**;**

**}**

void 2Bubble\_Sort**(**int **\***num**,** int n**)**

**{**

int i**,** j**;**

**for(**i **=** 0**;** i **<** n**;** i**++)**

**{**

**for(**j **=** 0**;** i **+** j **<** n **-** 1**;** j**++)**

**{**

**if(**num**[**j**]** **>** num**[**j **+** 1**])**

**{**

int temp **=** num**[**j**];**

num**[**j**]** **=** num**[**j **+** 1**];**

num**[**j **+** 1**]** **=** temp**;**

**}**

Print**(**num**,** n**);**

**}**

**}**

**return;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*杨辉三角\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

3main**()** **{**

int i**,**j**,**n**=**0**,**a**[**17**][**17**]=** **{**0**};**

**while(**n**<**1 **||** n**>**16**)** **{**

printf**(**"请输入杨辉三角形的行数:"**);**

scanf**(**"%d"**,&**n**);**

**}**