**编译原理课程设计**

学 院（系）： 电子信息与电气工程学部

学 生 姓 名： 王志洋

学 号： 201785004

班 级： 电计1701

同 组 人 ： 袁逸夫

李家平

大连理工大学

Dalian University of Technology

目 录

[1 课程设计基本情况 1](#_Toc27683405)

[1.1 样本语言文法定义 4](#_Toc27683406)

[1.2 单词的识别模型-有穷自动机DFA 6](#_Toc27683407)

[1.2.1 删除注释的DFA 6](#_Toc27683408)

[1.2.2 词法分析使用的DFA 6](#_Toc27683409)

[1.3 设计及实现功能概述 7](#_Toc27683410)

[1.4 小组任务分配 7](#_Toc27683411)

[2 词法分析程序的实现 4](#_Toc27683412)

[2.1 词法分析实现功能 8](#_Toc27683413)

[2.2 词法分析创新点 8](#_Toc27683414)

[2.3 词法分析测试结果 9](#_Toc27683415)

[2.3.1 正确测试样例 9](#_Toc27683416)

[2.3.2 错误测试样例 9](#_Toc27683417)

[3 语法分析程序的实现 10](#_Toc27683418)

[3.1 语法分析实现功能 10](#_Toc27683419)

[3.2 程序流程分析 11](#_Toc27683420)

[3.3 关键算法分析 11](#_Toc27683421)

[3.4 输出结果 13](#_Toc27683422)

[4 语义分析的实现 13](#_Toc27683423)

[4.1 功能简述 14](#_Toc27683424)

[4.2 程序流程分析 14](#_Toc27683424)

[4.2.1 布尔表达式和运算表达式生成四元式 15](#_Toc27683416)

[4.2.2 if语句生成四元式 16](#_Toc27683417)

[4.2.3 while语句生成四元式 17](#_Toc27683416)

[4.2.4 for语句生成四元式 18](#_Toc27683417)

[4.3 算法说明 18](#_Toc27683424)

[4.2.1 主要数据结构声明 19](#_Toc27683416)

[4.2.2 核心类（函数）说明 19](#_Toc27683417)

[4.4 测试结果举例 20](#_Toc27683424)

[4.5 语法、语义错误分析 20](#_Toc27683424)

[4.5.1 正确测试样例 21](#_Toc27683416)

[4.5.2 错误测试样例 22](#_Toc27683417)

[5 解释程序的实现 2](#_Toc27683425)

[5.1 功能简述 23](#_Toc27683424)

[5.2 算法说明 23](#_Toc27683424)

[5.2.1 主要数据结构声明 23](#_Toc27683416)

[5.2.2 核心类（函数）说明 24](#_Toc27683417)

[5.3 测试结果举例 25](#_Toc27683424)

[6 GUI界面实现 25](#_Toc27683426)

[7 运行正确程序样例展示 25](#_Toc27683427)

[7.1 斐波那契数列及求阶乘 25](#_Toc27683428)

[7.2 算术优先级 26](#_Toc27683429)

[感 想 26](#_Toc27683430)

[附录 程序报错项目表 28](#_Toc27683431)

# 1 课程设计基本情况

## 1.1 样本语言文法定义

源语言实现的为C语言的子集，基本文法（LL（1）文法）定义如下：

（1）主控语句 用于进入主语句体等

Z->gY|@

Y->3()S

S->{AC4b;}|@

（2）子程序语句 用于函数调用

Y->b(X)SgY

X->gbW

W->,X|@

（3）数据定义 满足先声明后使用规则

A->BA|@

B->gb;

（4）算术表达式 加入+，-，\*，/优先级判断

T->+OT|-OT|@

O->PU

U->\*PU|/PU|@

P->a|b8|(N)

8->(V)|@

V->b9

9->,V|@

（5）if else for while 及赋值语句定义 支持控制语句的相互嵌套

C->DC|@

D->E|F|G|I|H|J|L|;

E->c(M)DQ

Q->dD

F->f(M)D

G->e(K;M;K)D

H->hN;

I->ib;

J->{C}

K->b=N

L->K;

（6）布尔表达式及逻辑表达式定义 并支持表达式之间的逻辑、布尔运算

M->jk

j->NR

k->6jk|5jk|@

N->OT

R->>N|<N|wN|xN|yN|zN

## 1.2 单词的识别模型-有穷自动机DFA

### 1.2.1 删除注释的DFA

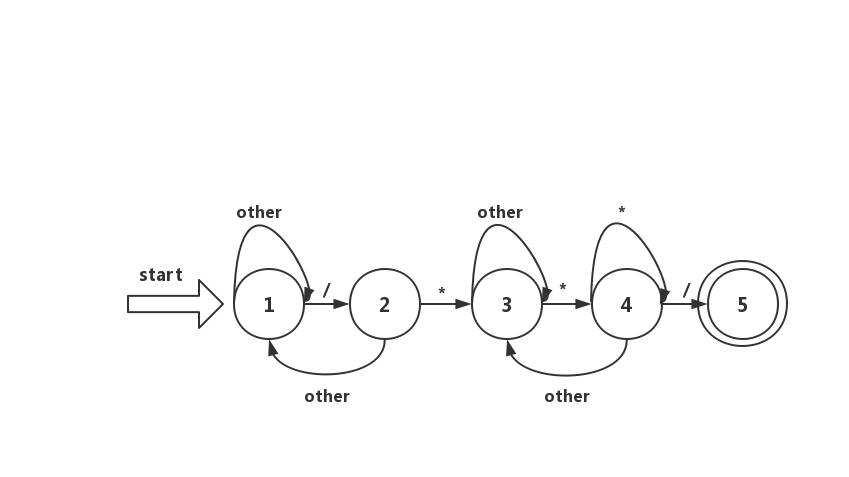


图1.1 删除注释的DFA

### 1.2.2 词法分析使用的DFA

在最终版代码中，我们小组对之前的代码进行了很多词法扩充，所以在最终版提交代码中，我们使用了基于正则表达式匹配的词法分析规则，具体的功能实现和详细的匹配规则见后文第2章节。

## 1.3 设计及实现功能概述

**开发语言：**python3.7

**开发环境：**pycharm2017.1

**文件结构：**lexical.py：词法分析，生成token表

syntax.py：语法分析，分析语法规则，生成LL（1）分析表

semantic.py：语义分析，递归下降子程序分析，建立符号表，分析属性文法，输出四元式

vm.py：根据四元式和符号表解释执行程序，输出程序运行结果

compile\_GUI.py：GUI界面，基于Tkinter图形库的图形界面，用于人机交互

**实现功能概述：**

（1）词法分析器：最终使用了基于正则表达式匹配的词法分析功能，可以实现对标识符、常数、类型符、运算符、界符、逻辑运算符等类别的区分，C语言全集的词法识别，并实现了注释过滤和词法分析报错功能（支持同时报多个错误）。

（2）语法分析器：在该程序中语法分析主要是分析给定文法是否为LL（1）文法并自动生成LL（1）预测分析表，方便后续语义分析使用求得First集，Follow集等。

（3）语义分析器：主体使用递归子程序分析法，实现了主程序语句，声明语句，控制语句及相互嵌套，算术表达式、逻辑表达式、布尔表达式，子函数调用，自增自减运算等功能，并对错误程序有完善的错误分析功能，最终生成符号表和四元式。

（4）翻译程序：根据四元式和符号表解释执行程序，输出程序的运行结果。

（5）界面：使用Tkinter图形库实现了简单的交互界面。

## 1.4 小组任务分配

本次实验中，我主要负责完成语法分析的大框工作，语义分析的大框工作，虚拟机设计三部分，辅助完成词法定义。

袁逸夫同学负责词法定义，语义和语法的错误分析，布尔运算的语义和语法设计，可视化界面编程。

李家平同学负责完成函数调用的语法和语义设计，参与算法设计。

# 2 词法分析程序的实现

## 2.1 词法分析实现功能

（1）从指定的文件路径打开文件，获取待处理的源代码

（2）替换’\n’, ’\r’, ‘\t’等无用字符

（3）删除注释和多余空格

（4）根据正则表达式字典逐行匹配识别，生成Token表（三元组）

（5）将获得的Token表存入Result.txt文件夹

词法分析在对源代码文件进行过滤后，从头到尾逐行扫描源程序中的字符，根据正则表达式匹配识别具体词法类别，生成Token表（格式：词法类别 具体字串 行号），并写入文件。同时增加错误处理功能，可以识别词法匹配错误和注释格式错误等问题，并进行报错。

## 2.2 词法分析创新点

采用了基于正则表达式匹配的词法分析识别手段，和DFA自动机识别不同的是，通过恰当可以很方便的对源语言的词法进行定义和扩展，例如：变量名定义为字母或下划线开头，之后附带任意个字母，数字，下划线。可以实现对C语言全集的词法分析。

同时，增加了错误处理功能，对无法识别的字符和注释不匹配等情况进行报错，并且支持多行报错。

## 2.3 词法分析测试结果

### 2.3.1 正确测试样例

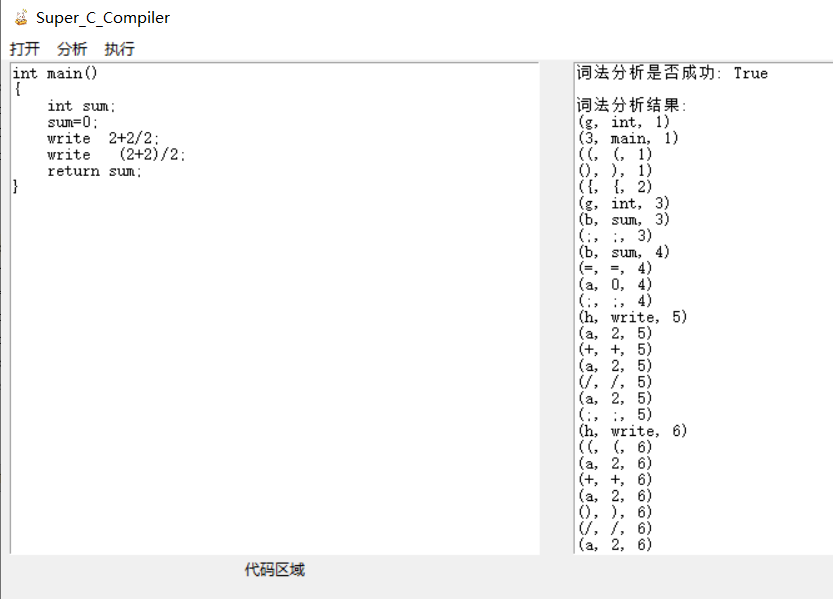
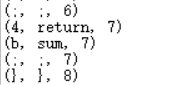
 

图2.1 词法分析正确样例（输出Token表）

以（g， int， 1）为例，g为词法分类，int为字串名称，1为行号。

### 2.3.2 错误测试样例

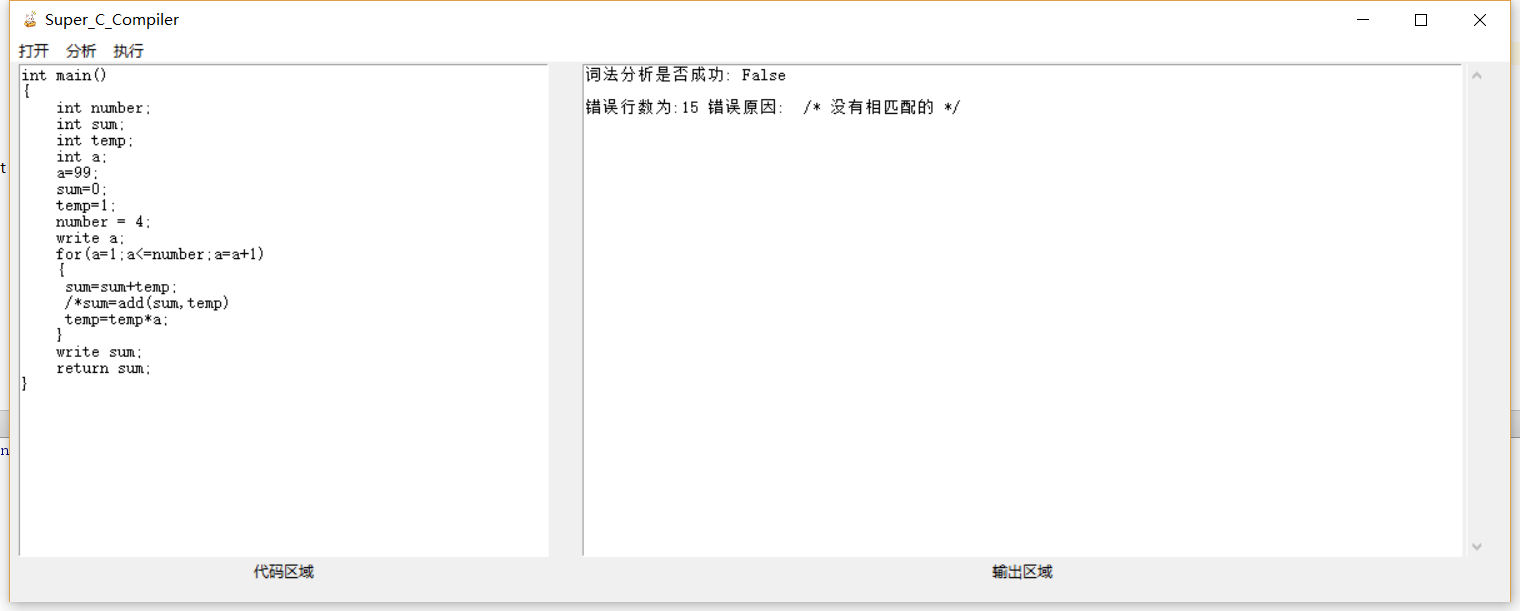


图2.2 词法分析错误样例1

该错误样例在第15行使用了注释/\*，却没有对应的\*/匹配。

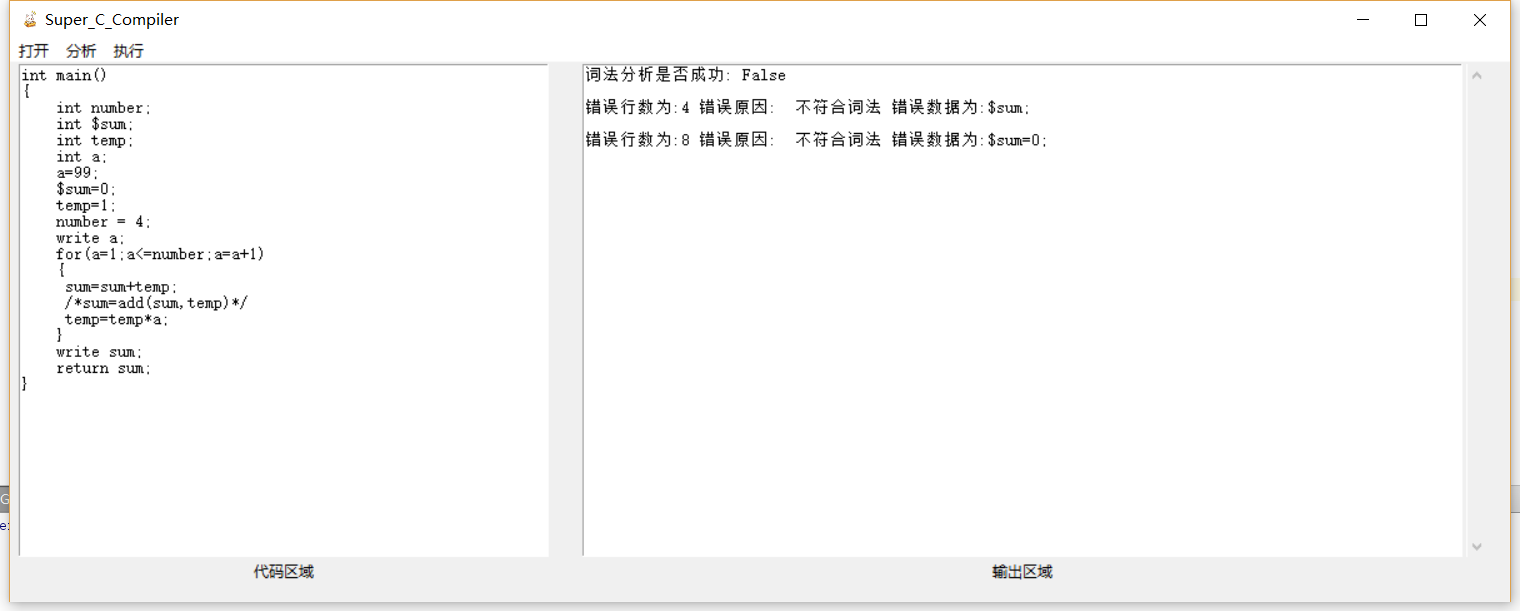


图2.3 词法分析错误样例2

该错误样例在第4行使用了错误的变量名$sum，不符合词法定义。

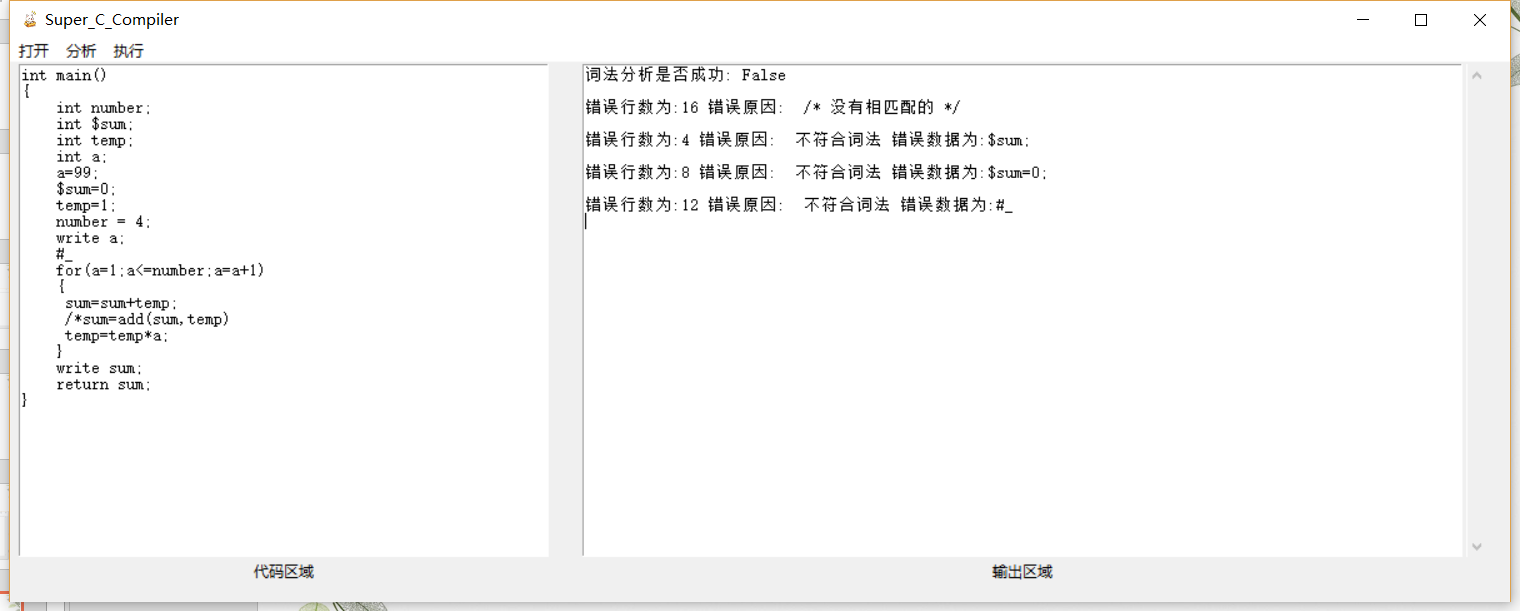


图2.4 词法分析错误样例3

展示可支持多行报错功能。

# 3 语法分析程序的实现

## 3.1 语法分析实现功能

为给定的文法求FIRST集和FOLLOW集，自动构造预测分析表。

若分析过程中发现给定文法不满足LL（1）规则，则输出文法冲突位置，便于改进。 并根据LL（1）表设计递归下降子程序，完成设计。



图3.1 输出文法冲突位置

## 3.2 程序流程分析

（1）读取文法文件Grammar.txt，并将诸如Z->gY|@的文法分割成多条。

（2） 将文法划分终结符，非终结符，起始符号，求每个非终结符的FIRST集和FOLLOW集。

（3）为给定文法自动构造预测分析表，并输出冲突位置。

（4）按照定义的文法，顺序构造递归子程序。当遇到空串时，下一步若读取到空串所在表达式的非终结符的FOLLOW集，则返回上一层。

## 3.3 关键算法分析

FIRST集构建：

（1）、如果 X是一个终结符号，那么 FIRST（X）=X;

（2）、如果 X是一个非终结符号，且X->Y12…Yn 是一个产生式，其中n>=1 ，那么如果对于某个 i，Y1Y2…Y(i-1)->@,且a属于FIRST(Yi),则将a加入FIRTST（X）中，若@属于FIRST（ Yi ）中且i属于{1，n},将@加入FIRTST（X）

（3）、 如果 X->@是一个产生式，那么将@ 加入 FIRST(X).

FOLLOW集：

1. 首先将$放入FOLLOW(Z)中.
2. 如果存在产生式A-> ，那么FIRST()中除了@的所有符号都在FOLLOW(B)中.
3. 如果存在一个产生式A->，或存在产生式 A->且 FIRST()包含 @，那么 FOLLOW(A)中的所有符号都在FOLLOW(B) 中

下面给出自动构造预测分析表的伪代码：

input: 文法G  
output：文法G的预测分析表M

for(文法G的每一个产生式A -> a)

{

for(每个终结符号a ϵ FIRST(a)){ 把A -> a加入M[A, a]中;}

if(@ ϵ FIRST(a))

{

for(每个b ϵ FOLLOW(A)){把A -> a加入M[A, a]中;}

}

}

for (所有无定义的表项M[A, a] ) 标上错误标志。

即通过该算法，在求得FIRST集和FOLLOW集的前提下，可以方便的自动求得LL（1）预测分析表，并对冲突的部分进行报错。

下面给出部分递归下降子程序的代码：

def \_S(self):

print(self.ch)

if self.ch.symbol == '{':

self.getNextch()

self.\_A()

self.\_C()

if self.ch.symbol == '4':

self.getNextch()

if self.ch.symbol == 'b':

self.val = self.ch.value

self.getNextch()

if self.ch.symbol == ';':

self.getNextch()

if self.ch.symbol == '}':

# 栈空，不读取下一个字符

pass

else:

# print("lose }")

self.\_err(self.ch.line, 'LOST-', 'Lost }')

else:

# print("lose ;")

self.\_err(self.ch.line, 'LOST-', 'Lost ;')

else:

# print("lose variable")

self.\_err(self.ch.line, 'TYPE-', 'Wrong return type')

else:

# print("lose return")

self.\_err(self.ch.line, 'LOST-', 'Lost return')

elif self.FOLLOW('S', self.ch.symbol):

return

else:

# print("lose {")

self.\_err(self.ch.line, 'LOST-', 'Lost {')

## 3.4 输出结果

输出LL（1）预测分析表。

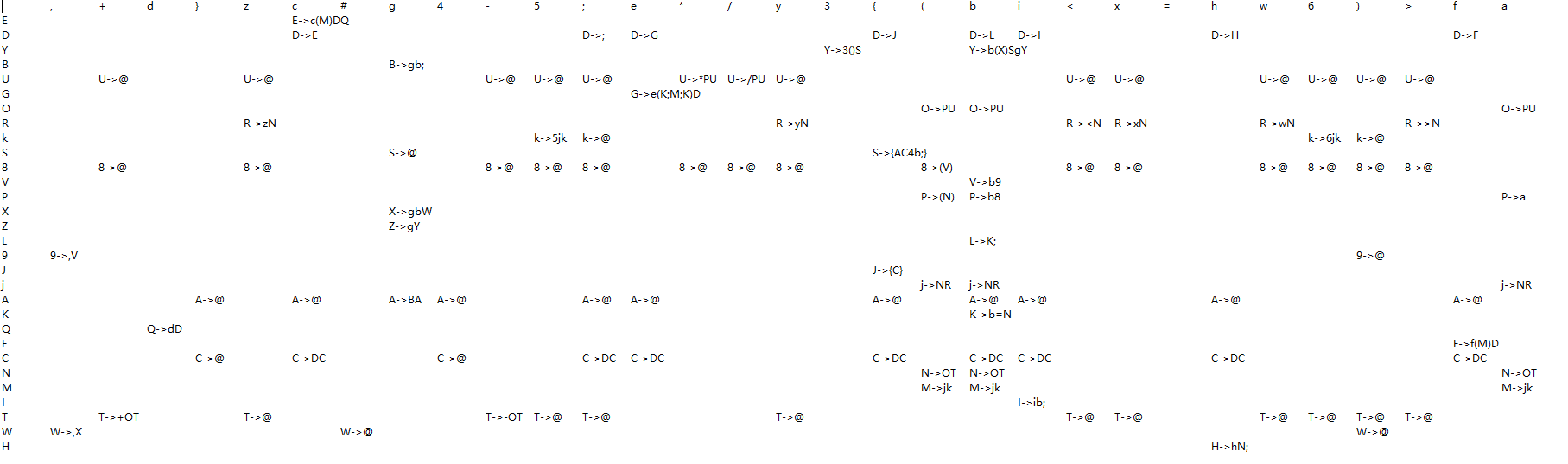


图3.2 LL（1）预测分析表

# 4 语义分析的实现

## 4.1 功能简述

在本次的课程设计中，我们将语义分析和递归子程序结合在一起，最终生成四元式代码，并在语义分析的同时，对符号表管理，创建了临时变量表。从而为之后的解释程序做好准备。

生成的四元式代码格式为：（OP，arg1，arg2，result）

其中 OP 是运算符；arg1 和 arg2 分别是第一和第二运算对象（当OP是单目运算符是，常常将运算对象定义为 arg1，\_,）；result 是编译程序为存放中间运算结果而临时引进的变量，或称为临时变量，我们将其保存在临时变量表中，用T0到Tn来表示。

由于控制语句的四元式代码涉及到跳转，所以我们在设计的时候利用了拉链回填法，即控制语句下一步跳转到的四元式标号设置为label(n)，当程序到达所应跳转的语句体时，将该语句的标号赋值给缺省的标号label(n)。

在本次的课程设计中，我们对while , for , if else , 等控制语句进行了四元式生成设计，并对布尔表达式和运算表达式进行了四元式设计。

## 4.2 程序流程分析

### 4.2.1 布尔表达式和运算表达式生成四元式

在设计中，为实现如（arg1,op,arg2）所示运算的四元式的产生，实际上是按照文法规则依次对表达式arg1, arg2进行读取处理操作。首先依次读取tokens流中弹出的数据，并从符号表中获得表达式arg1，arg2的值，并产生临时变量Tn , 最终和op操作符生成相应的四元式。

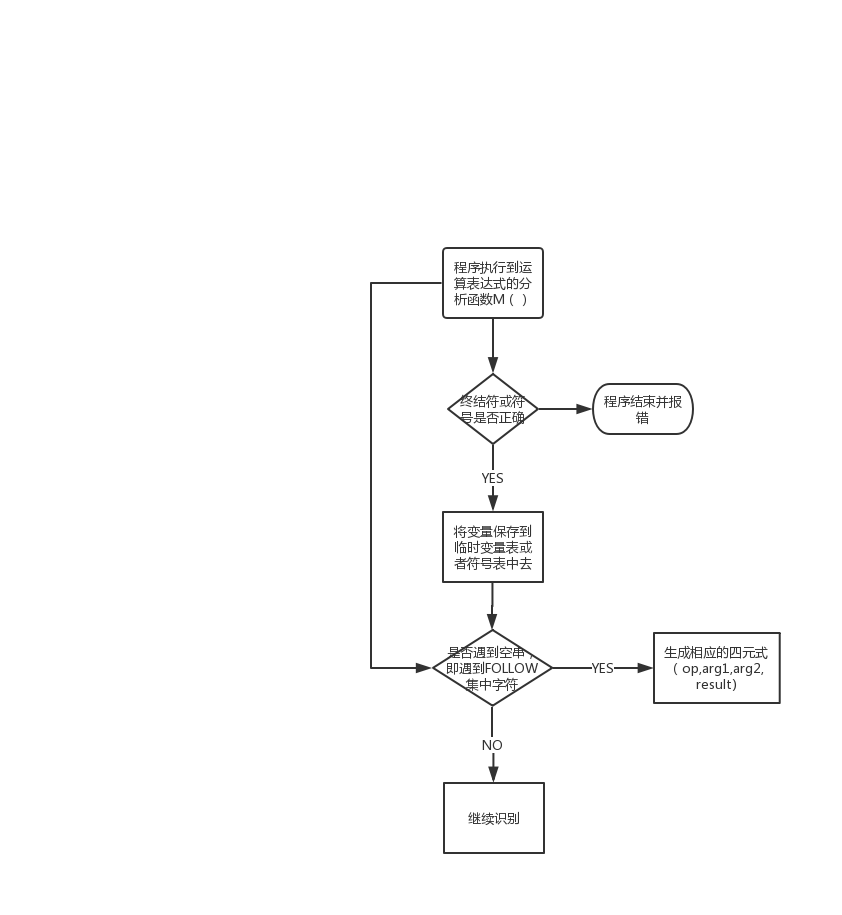


图 4.1 运算表达式和布尔表达式生成四元式流程图

### 4.2.2 if语句生成四元式

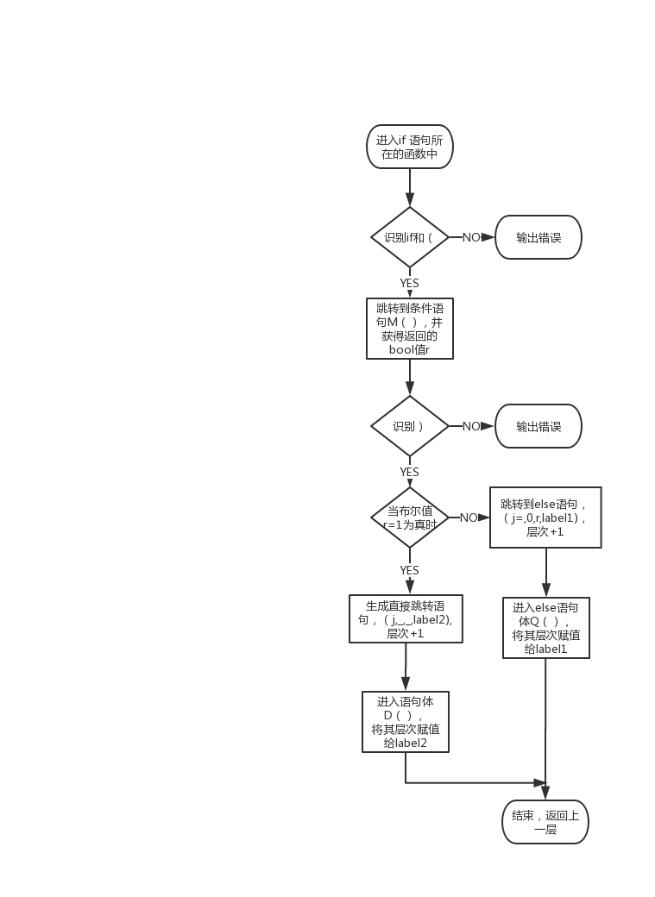


图 4.2 if 语句生成四元式流程图

### 4.2.3 while语句生成四元式

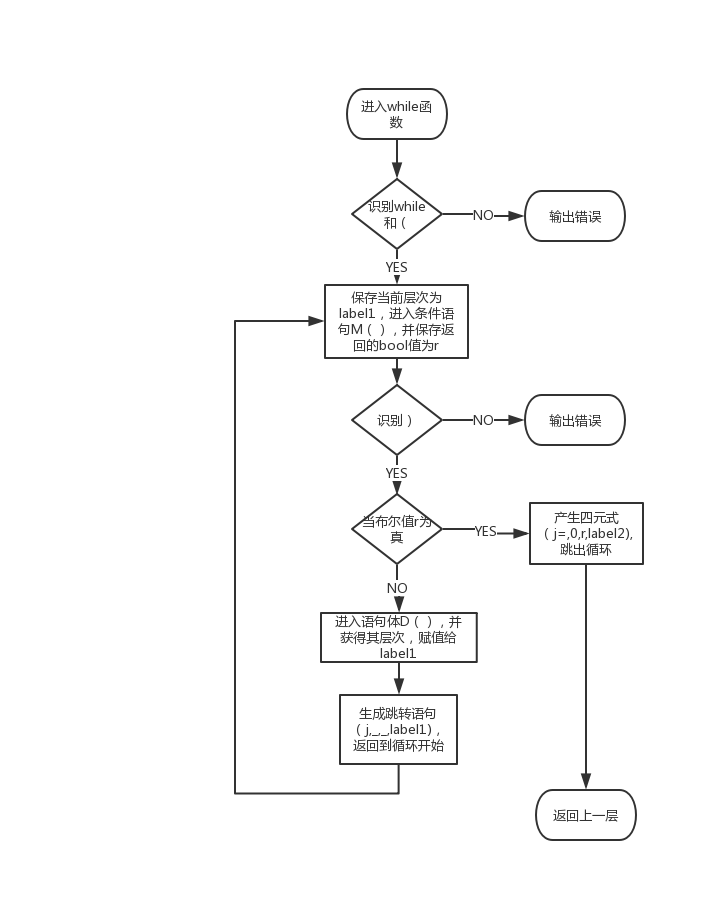


图 4.3 while 语句生成四元式流程图

### 4.2.3 for语句生成四元式

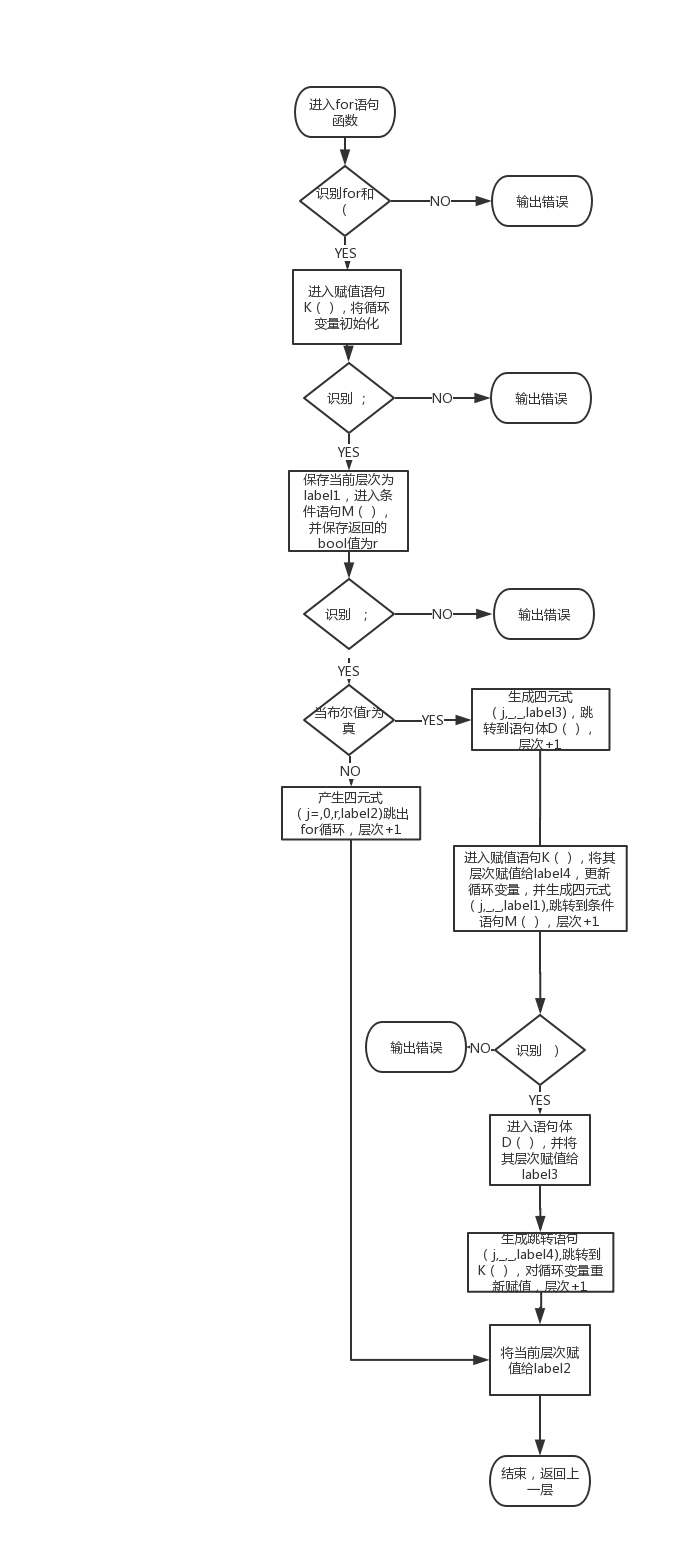


图 4.4 for 语句生成四元式流程图

## 4.3 算法说明

### 4.3.1 主要数据结构声明

def createSymbolList(self, input\_str, map\_list, map\_line):

# 建立符号表

# 词法分析给出的所有元素

self.list = []

for i, ch in enumerate(input\_str):

self.list.append(element(ch, map\_list[i],map\_line[i]))

# 符号表

self.chart = dict()

# 函数名表

self.function =dict()

# 四元式表

self.seq\_list = list()

#层次数

self.seq\_num = 0

self.seq\_index=0

self.val=0

global newT\_num

newT\_num = 0

# 临时变量表

self.temp\_list = list()

class label():#标签定义

def \_\_init\_\_(self, value=None):

self.value = value

def \_\_repr\_\_(self):

return str(self.value)

def \_\_str\_\_(self):

return str(self.value)

### 4.3.2 核心类（函数）说明

Sequence()#自动生成四元式

NewT()#申请临时变量

element()#处理每一个符号的信息，如所在行数，值，状态

MyException()#处理错误

Pro()#递归子程序的具体过程

## 4.4 测试结果举例

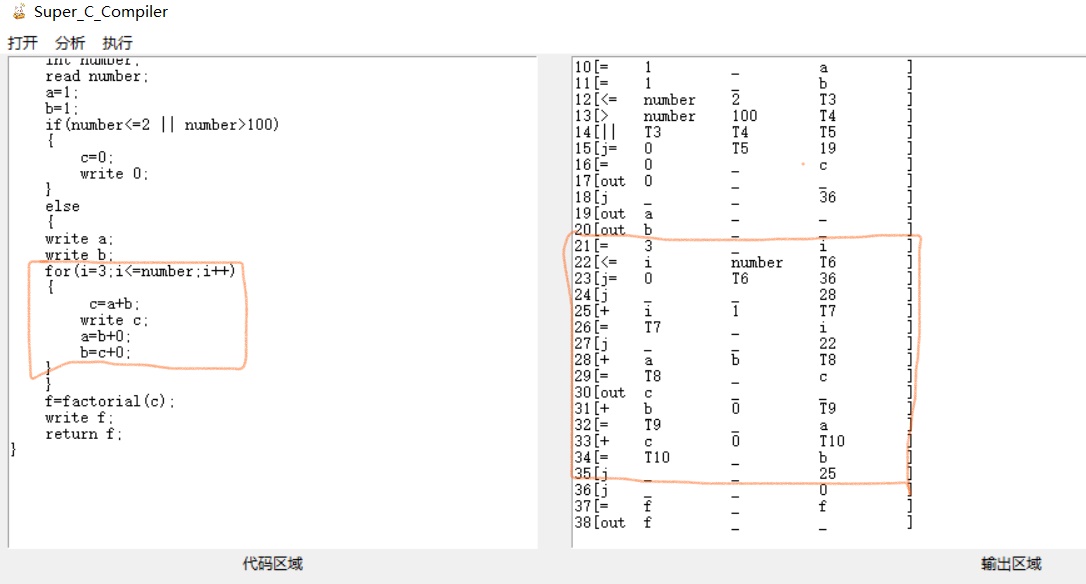


图 4.4 语义分析测试结果

分析上图可以看出，对于for语句，21行初始化i=3。22行为条件语句判断，并将结果保存在临时变量T6中，接下来若T6为假，则跳出循环到36行.反之则跳到语句体，即28—34行。语句体执行完后，跳转到25行，进行+1操作，并再次跳到22行判断是否正确。可以看出，实现了预期目标。

## 4.5 语法、语义错误分析

实现了4大类共20种小类的错误识别，详细的错误识别项目表见附录。同时实现了同一个程序多个报错的功能。在基础的递归下降子程序分析中，如果遇到错误就会跳出递归到错误处理部分，为了实现多个报错，基本思路为递归下降子程序跳转到错误分析部分时，对错误进行“修补”使其继续进行递归下降分析，即可实现识别多个错误的功能。

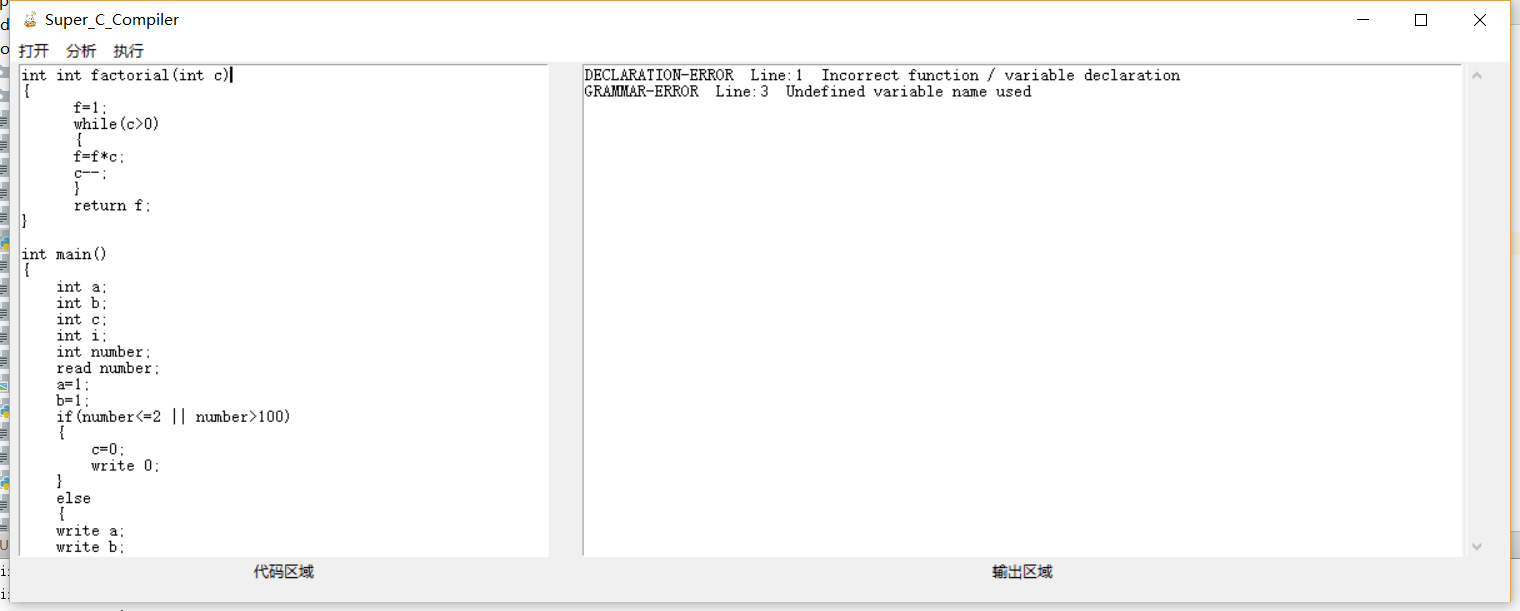
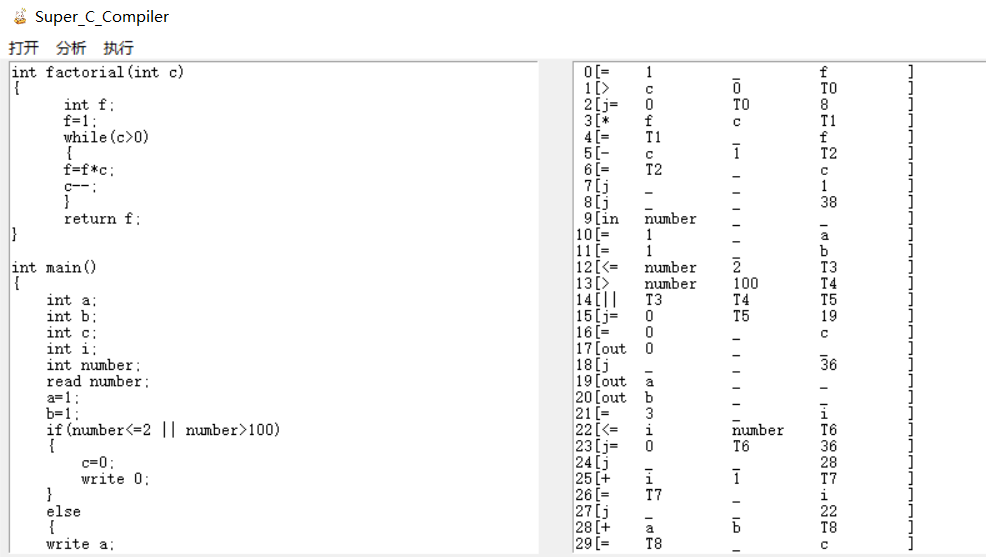


图4.1 同一程序多个错误

在此例中，第一行出现了不正确的函数定义的错误，程序识别第二个int并进入错误处理，程序对这个int的错误进行记录并忽略，继续向下读字符，第三行出现了函数未声明就使用的错误。如果是出现缺少‘｛’，‘（’等符号时，进行修补即可。

## 4.6 测试结果举例

### 4.6.1正确样例（生成四元式）



### 4.6.2 错误样例（不同报错）

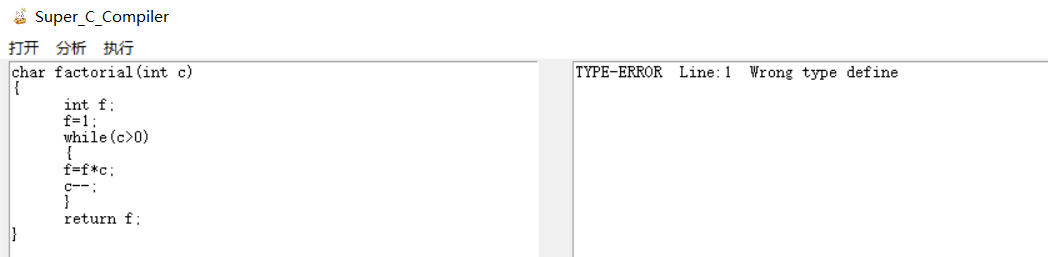


图4.1 函数声明类型错误

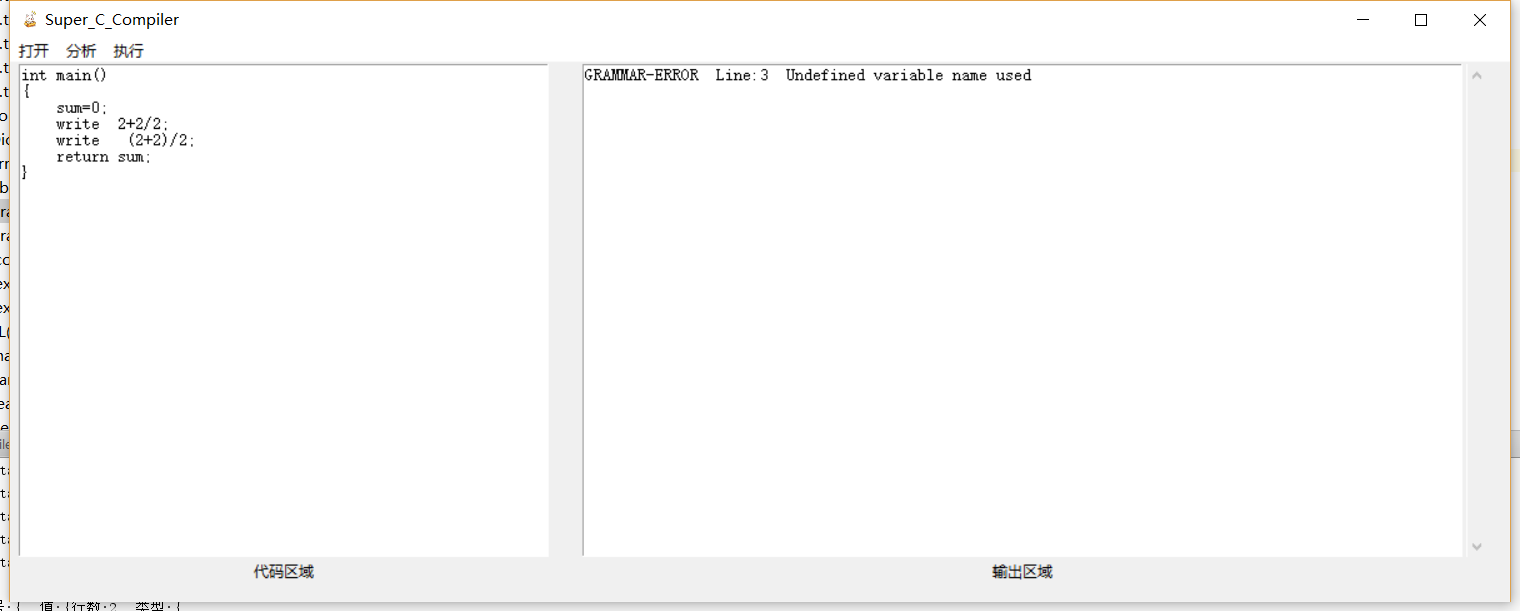
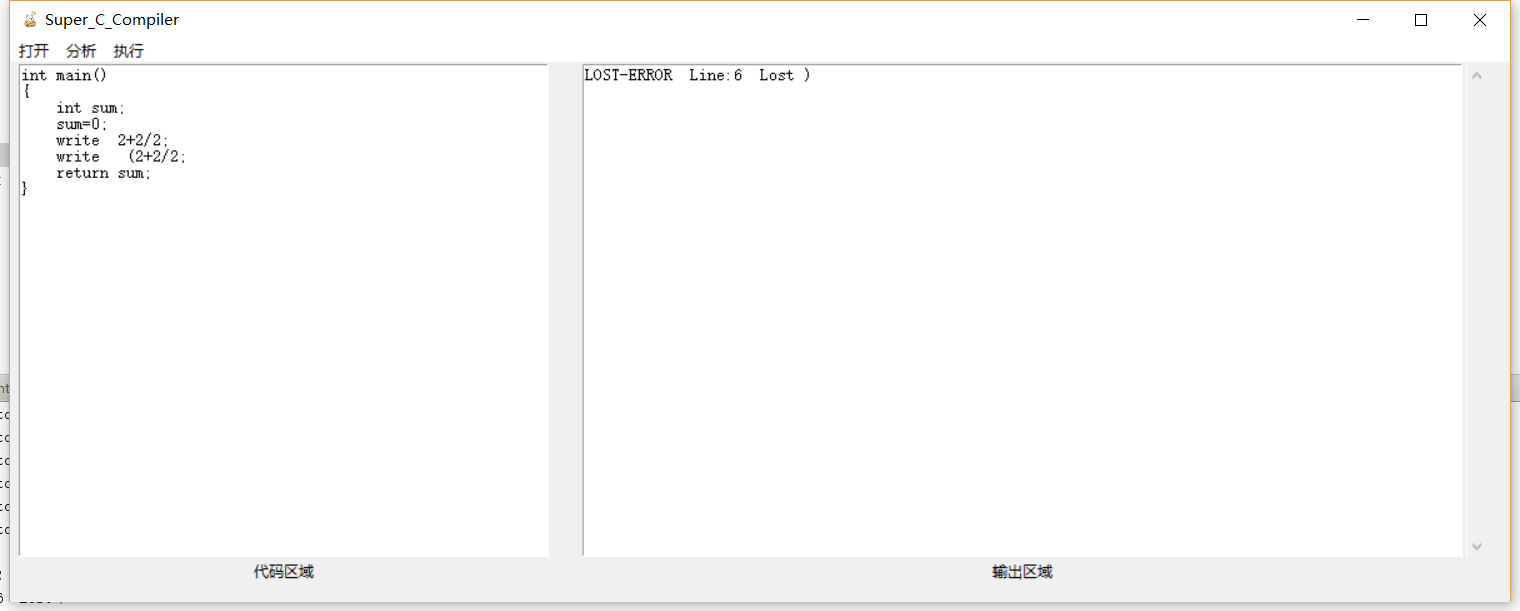


图4.2 变量未声明就使用



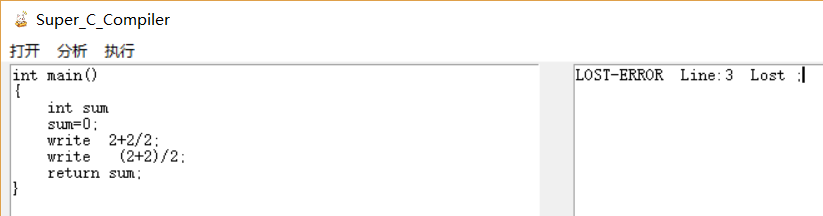


图4.2 缺少),;等符号

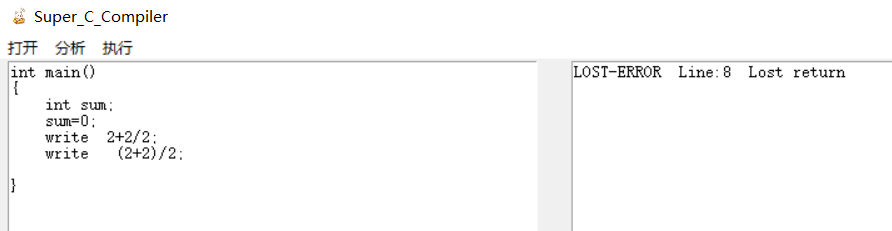


图4.2 缺少return



图4.2 if表达式格式错误

# 5 解释程序的实现

## 5.1 功能简述

此次课程设计当中，我们实现了解释程序，即将语义分析所生成的中间代码（四元 式）作为输入，通过分析函数实现了由计算机根据四元式的逻辑执行分析源程序代码， 并对于符号表进行相应的更新，最终输出符号表中的相关数据。

## 5.2 算法说明

### 5.2.1 主要数据结构声明

p.chart = chart # 符号表 类型：dict

p.temp\_list = newT # 临时变量表 类型：NewT

p.seq\_list = sequence # 四元式表 类型：Sequence

p.seq\_index=seq\_index #主函数四元式开始的位置

### 5.2.1 主要函数声明

find\_value(name):#查找符号表数值

VM(chart,newT,sequence,seq\_index)：

//最核心的函数，根据四元式来执行算数逻辑上的运算

## 5.3 测试结果

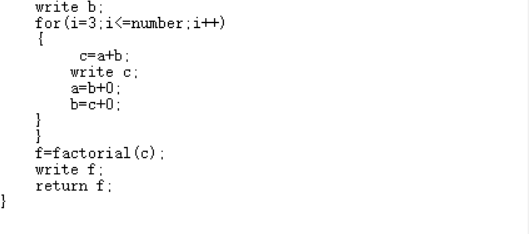


图 5.3 解释程序测试结果

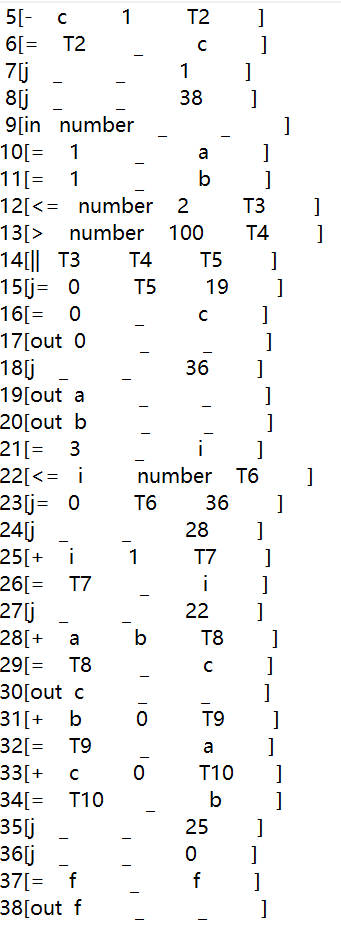
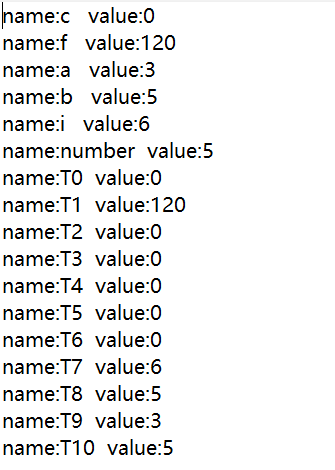
 

图 5.4 四元式代码以及在符号表中的值

对于四元式的操作根据具体的符号通过 if 的语句来进一步作出跳转动作（修改跳转的行数 index 的值），并对符号表进行读取存储操作。

动作有以下几种：

（1）当是赋值语句以及算数运算语句时，做完相应的操作，index++；

（2）当是 bool 语句的跳转操作时，先计算 bool 式结果，根据结果跳转：

a)如果为真，则跳转到 result 所对应的行号，即index = item.result.value

b)如果是假，则 index ++

（3）当时输入输出操作时，根据动作将值保存或者输出

# 6 GUI界面实现

基于Tkinter图形库，使用python语言简单实现了界面，其中左边是源代码窗口，实时显示打开的源代码。右边是输出窗口，显示词法、语法、语义的输出结果及最后程序的运行结果。左上角菜单栏“打开”功能可以检索打开对应的文件，“分析”功能可以分别输出词法分析、语法分析、语义分析结果，“执行”功能为运行解释程序，输出代码执行结果。

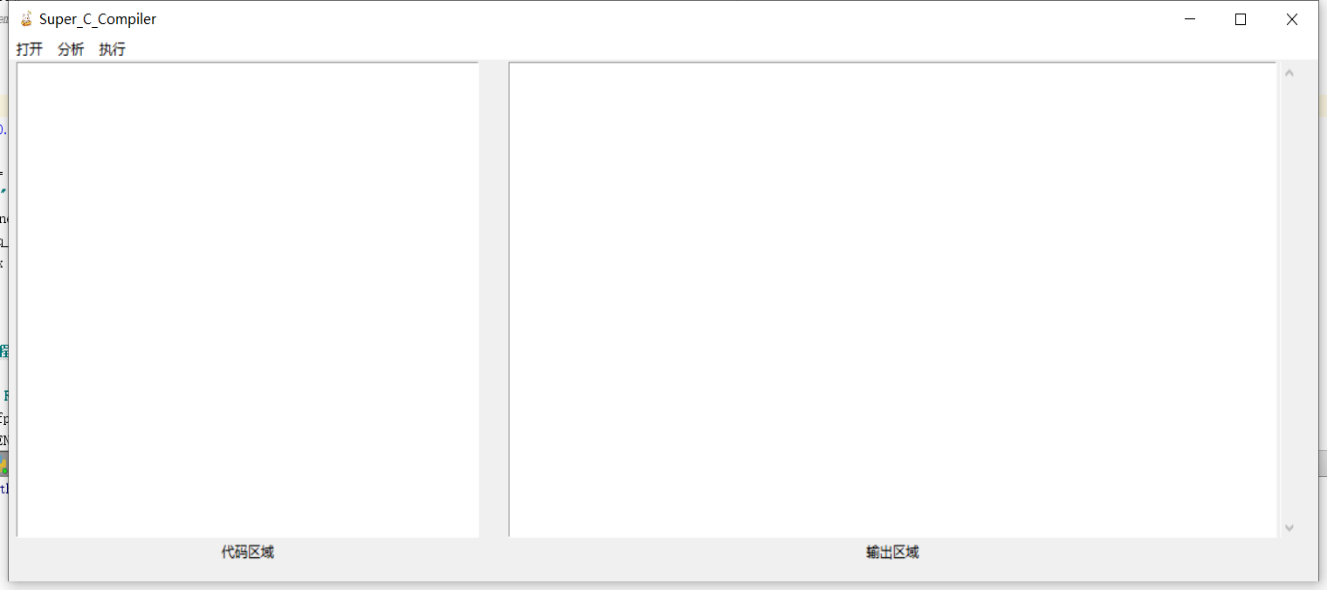


图6.1 界面展示

# 7 运行正确程序样例展示

## 7.1 斐波那契数列及求阶乘

该程序输出n个数的斐波那契数列并计算第n个数的阶乘。其中输入的值为5，得出前5项的斐波那契数列为1，1，2，3，5，并且5的阶乘为120，符合程序输出。

其中展示了函数调用，自增自减运算，布尔表达式，逻辑表达式，if，else，for，while嵌套调用等功能。



图7.1 斐波那契数列程序运行结果

## 7.2 算术优先级

一个简单的算术程序，展示了算术优先级高低。



图7.2 算术优先级程序运行结果

# 感 想

首先就是感觉时间很快，三周转瞬而过，做的越多，就越感到知识的宽广。

还记得最开始上课时，老师展示的上届同学的作品，那时感觉真厉害，能实现一个小型编译器，当时真的感觉好神奇。不知不觉间，自己也通过不断地学习，真正的实现了一个小型编译器。

很荣幸能够和袁逸夫同学，李家平同学一起完成这个作品，并被推举为组长，统筹整个流程，我也对整个编译器的架构有了更深的了解。此次课程设计共分为词法分析，语法分析，语义分析，虚拟机分析，图形界面五部分功能。本次实验中，我主要负责完成语法分析的大框工作，语义分析的大框设计，虚拟机设计三部分，辅助完成词法定义，其它附加功能则由另外两位同学完成。

由于编译器设计中对于数据的访问查找工作非常多，利用C语言的数组进行存储会增加查找的工作量，不够简洁。所以我们采用python开发，在接下来我负责的设计中，我主要利用python中的字典功能，可以通过键——值的映射关系，更加方便查找。袁逸夫同学之前已经以用DFA设计好了词法程序，而我在完成负责的语义工作时，由于需要不断扩充增加功能以及标志符，DFA不易改变，经查阅龙书以及教辅，所以我在他的的思路上，进行修改，通过建立正则表达式字典，达到快速匹配以及扩充的作用。在实验中，我通过实验进一步认识到了正则表达式和DFA的等价关系。

在语法分析时，我采用了递归子程序法，但是由于随着文法条数的增加，手工计算FIRST,FOLLOW集非常浪费时间，且不易判断文法是否符合LL（1）文法，确实让我在设计递归子程序时很头痛。后来在翻阅教材和龙书时发现可以自动机器求取FIRST，FOLLOW集，根据伪代码定义，我最终实现了一个可以自动生成LL（1）预测分析表，以及FIRST,FOLLOW集的程序。为之后的语法分析起了很大作用。（对于想偷懒的我来说很好用哈哈）

根据上面求得的FIRST,FOLLOW集，按照文法定义便很容易的实现了语法功能扩充，语法上我主要完成了基本语法功能，如条件表达式，算术表达式，控制语句，语句嵌套等功能。而复杂功能函数调用则由李家平同学补充完成，布尔表达式则有袁逸夫同学完成。

语义分析功能实现时，在设计时我遇到的难点主要是for语句，由于跳转比较复杂，有四个跳转，生成的四元式或者是一直在错误的位置回填，或者就是顺序逻辑不清 。经过不断改进试错，最终还是实现了for语句，有一种柳暗花明的感觉。而函数调用以及布尔表达式的语义设计同样由另两位同学完成。

最终解释阶段，本打算生成可执行exe文件，但是由于需要对数据进行寄存器层次的运算和访问，由于时间关系，没有来的及实现，只能仓促的完成了一个虚拟机代码，通过对四元式进行按行处理，并访问符号表，最终得到结果。算是留下了一个遗憾。

最后，我要特别感谢姚卫红老师的教导，姚老师讲课非常细致耐心，让我打下一个比较好的理论基础，才能让我更加自信的完成这个编译器的设计，我今后也会更加努力的学习专业知识，做一个合格的程序员！

# 附录 程序报错项目表

**词法分析：**

1. “不符合词法”
2. “错误的变量名定义”
3. “/\*没有相匹配的\*/”

**语法、语义分析：**

**TYPE-ERROR**

1. Wrong type define
2. Wrong return type
3. Wrong type declaration

**GRAMMAR-ERROR**

1. Wrong Expression
2. Wrong if expression
3. Wrong else expression
4. Wrong while expression
5. Wrong for expression
6. Wrong write expression
7. Wrong read expression
8. Undefined variable name used
9. Wrong operation expression
10. Wrong Logic expression

**LOSE-ERROR**

1. Lost ;
2. Lost }
3. Lost return
4. Lost {
5. Lost )
6. Lost (

**DECLARATION-ERROR**

1. Wrong type declaration
2. Incorrect function / variable declaration