**程 设 计 报 告**

设计题目：一个简单文法的解释器的设计与实现

**摘 要**

编译原理是计算机科学与技术专业一门重要的专业课, 它具有很强的理论性与实践性，目的是系统地向学生介绍编译系统的结构、工作原理以及编译程序各组成部分的设计原理和实现技术，在计算机本科教学中占有十分重要的地位。计算机语言之所以能由单一的机器语言发展到现今的数千种高级语言，就是因为有了编译技术。编译技术是计算机科学中发展得最迅速、最成熟的一个分支，它集中体现了计算机发展的成果与精华。本课设是词法分析、语法分析、语义分析的综合，外加上扩展任务中间代码的优化和目标代码的生成，主要是锻炼学生的逻辑思维能力，进一步理解编译原理的方法和步骤。

本次课程设计我们小组完成了一个简单文法编译器的设计与实现，其语言要求与C语言一致，我们参考了了C99标准，设计出了类C的简单但完整的程序生成文法，以及对应的翻译文法。在此基础上我们还做了处理赋值语句、if-else语句、while语句的语法分析和四元式的生成，这样我们就设计完了四元式的生成。然后，我们又设计了符号表，对应四元式的目标代码生成。

其中的特色点有：在词法、语法分析阶段能够检测出错误，并且能指出错误在哪一行，具体为什么错误；表示式的四元式采用了逆波兰式的方法；同时像if- while, 我们的编译器能判断其中的boolean表达式的真值，从而能采用正确的逻辑得出正确的结果；

**关键词**：编译原理，完整编译器，LL1分析法，语法制导，四元式，符号表，目标代码

**目 录**

**摘要** I

**1 概述** 6

**2 课程设计任务及要求** 8

2.1 设计任务8

2.2 设计要求8

**3 算法与数据结构** 9

3.1算法的总体思想（流程） 9

3.2 词法扫描模块 9

3.2.1 功能 9

3.2.2 数据结构 9

3.2.3 算法 11

3.3 语法分析模块 12

3.3.1功能 12

3.3.2 数据结构 12

3.3.3算法 13

3.3.4算法流程图 14

3.4 语义分析四元式以及符号表分析模块 20

3.4.1功能 20

3.4.2 数据结构 20

3.4.3算法 21

3.5目标代码生成模块 29

3.5.1功能 29

3.5.2 数据结构 29

3.5.3算法 32

**4. 程序设计与实现** 33

4.1 程序流程图 33

4.1.1程序流程图

4.1.2设计框架

4.1.3文法的设计

4.1.4翻译文法的设计

4.2 程序说明 33

4.2.1程序说明

4.2.2软件结构

4.3 实验结果 37

4.3.1测试用例

4.3.2用例语法正确时结果

4.3.3用例语法错误时结果

**5. 结论** 55

**6. 参考文献。** 56

**7. 收获、体会和建议。** 57

1. **概述**

编译原理是计算机专业的一门重要专业课，旨在介绍编译程序构造的一般原理和基本方法。内容包括语言和文法、词法分析、语法分析、语法制导翻译、中间代码生成、存储管理、代码优化和目标代码生成。编译原理是计算机专业设置的一门重要的专业课程。虽然只有少数人从事编译方面的工作，但是这门课在理论、技术、方法上都对学生提供了系统而有效的训练，有利于提高软件人员的素质和能力。由于时间和同学们的水平有限，故本次课设内容只涉及到了词法分析，语法分析，及语义分析中的中间代码的四元式生成和符号表以及目标代码的生成(仅在语义分析阶段做了简单的优化)。具体概述介如下：  
 1.词法分析：在这个阶段编译器实际阅读源程序（通常以分析程序字符流的形式表示）。扫描程序执行词法分析注释树符号表：它将字符序列收集到称作记号错误处的有意义单元中，记号同自然语言，与源代码理器语中的字词相似。因此可以认为扫描程序执行与优化程序拼写相似的任务。本实验的词法分析程序用于生成Token序列。并设置简单的出错处理，可以指出哪个单词在哪行出错并简单指出出错的原因  
 2.语法分析：该程序从扫描程序中获取记号形式的源代码，并完成定义程序结构的语法分析，这与自然语言中句子的语法分析类似。语法分析定义了程序的结构元素及其关系。其任务是识别和处理比单词更大的语法单位。本实验用于指出程序设计语言中的表达式、各种说明和语句乃至全部源程序其中的语法错误；必要时，可生成内部形式，便于下一阶段处理。  
 3.语义分析：程序的语义就是它的“意思”，它与语法或结构不同。程序的语义确定程序的运行，但是大多数的程序设计语言都具有在执行之前被确定而不易由语法表示和由分析程序分析的特征。这些特征被称作静态语义，而语义分析程序的任务就是分析这样的语义。一般的程序设计语言的典型静态语义包括声明和类型检查。由于同学们水平有限，我们只做了其中的符号表部分。

4.中间代码：根据中间代码的类型和优化的类型，该代码可以是文本串的数组、临时文本文件或是结构的连接列表。对于进行复杂优化的编译器。由于同学们水平有限，我们只做了四元式的生成。

5.目标代码：根据中间代码结合语义分析时得到的符号表再结合特地的目标机器生存目标代码(本次使用的是8086处理器的目标代码格式)。经过生成的目标代码由于一定的水平限制还不能在特地机器下跑起来，但是我们整体的方向是对的。

编译原理课程兼有很强的理论性和实践性，是计算机专业的一门非常重要的专业基础课程，在系统软件中占有十分重要的地位。编译原理课程设计是本课程重要的综合实践教学环节，是对平时实验的一个补充。通过编译器相关子系统的设计，使学生能够更好地掌握编译原理的基本理论和编译程序构造的基本方法和技巧，融会贯通本课程所学专业理论知识；培养学生独立分析问题、解决问题的能力，以及系统软件设计的能力；培养学生的创新能力及团队协作精神。

**2.课程设计任务及要求**

# 2.1 设计任务

在下列内容中任选其一：

1、一个简单文法的编译器前端的设计与实现。

2、一个简单文法的编译器后端的设计与实现。

3、一个简单文法的编译器的设计与实现。

4、自选一个感兴趣的与编译原理有关的问题加以实现，要求难度相当。

这次实验我们小组所做的内容涵盖了一个编译器的前端和后端，因此基本实现了一个简单文法的编译器的设计与实现。

主要内容如下：

1. 定义一个简单程序设计语言文法（包括变量说明语句、算术运算表达式、赋值语句；扩展包括逻辑运算表达式、If语句、While语句等）。
2. 词法扫描设计实现。
3. 语法分析设计实现。
4. 中间代码设计实现。
5. 符号表的生成。
6. 目标汇编语言的生成。

## 2.2 设计要求

1、在深入理解编译原理基本原理的基础上，对于选定的题目，以小组为单位，先确定设计方案；

2、设计系统的数据结构和程序结构，设计每个模块的处理流程。要求设计合理；

3、编程序实现系统，要求实现可视化的运行界面，界面应清楚地反映出系统的运行结果；

4、确定测试方案，选择测试用例，对系统进行测试；

5、运行系统并要通过验收，讲解运行结果，说明系统的特色和创新之处，并回答指导教师的提问；

6、提交课程设计报告。

**3 算法与数据结构**

**3.1算法的总体思想**

我们的课程设计实验基于自动机的词法扫描，采用LL1分析法的语法分析的方法实现编译器前端和对应后端代码的生成(目标机器为X8086)。整体框架主要部分有词法分析、语法分析、语义分析中四元式生成、符号表建立等以及对应程序的目标代码生成(运用单寄存器下目标生成算法，将四元式翻译成为汇编语言。)。在搭建框架的时候我们定义了一套接口，以协同合作。

整体的框架类图关系如下(使用UML组织结构,其中也可以看出我们的接口之间的连接关系)：

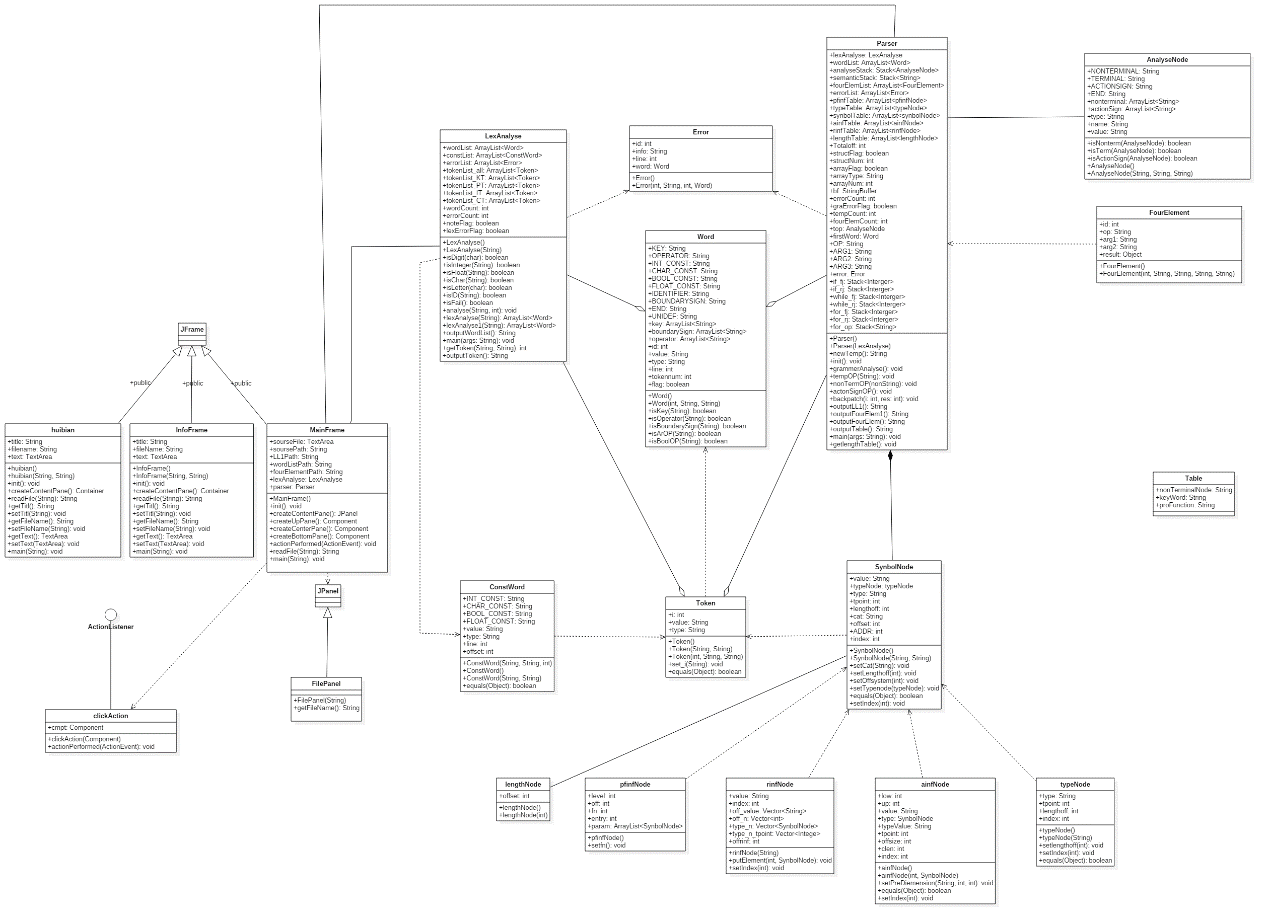


图3-1程序框架图

**3.2 词法分析**

3.2.1 功能

词法分析程序又称扫描器，任务有二：

1. 识别单词——从用户的源程序中把单词分离出来；
2. 翻译单词——把单词转换成机内表示，便于后续处理。

词法分析是编译的第一步，其目的是对程序进行扫描，并生成token序列，以便后面进行语法分析。

3.2.2 数据结构

词法分析中用到的数据结构如下所示：

LexAnalyse词法分析器类设计:



图3-2LexAnalyse词法分析器类结构

在程序中用链表存储总Token表，关键字表，界符表，常数表，标识符表以及错误信息表。对应的结点内容如下所列。

Token类设计:

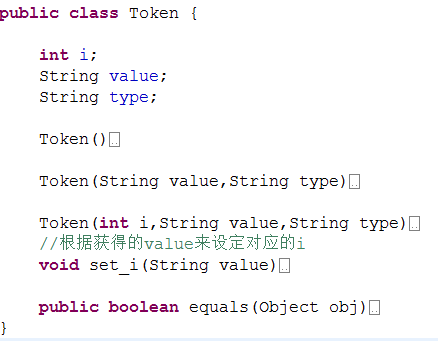


图3-3Token类设计

ConstWord类设计:

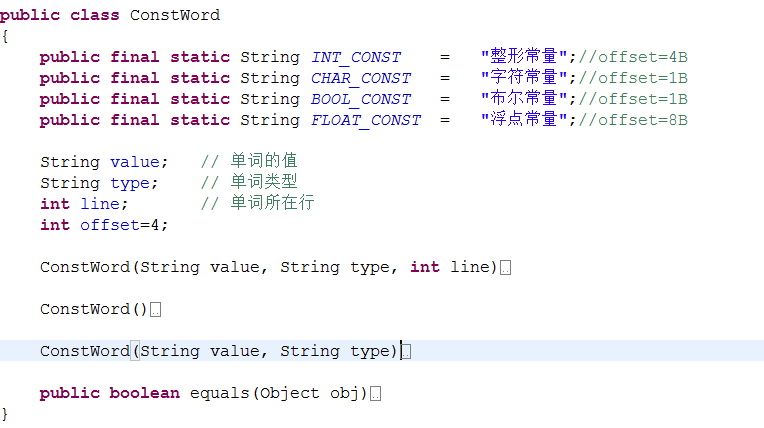


图3-4 ConstWord类设计

Word类设计:



图3-5Word类设计

Error类设计:

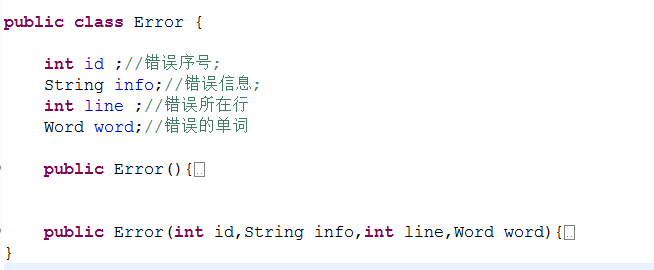


图3-6 Error类设计:

关键字定义表如下:

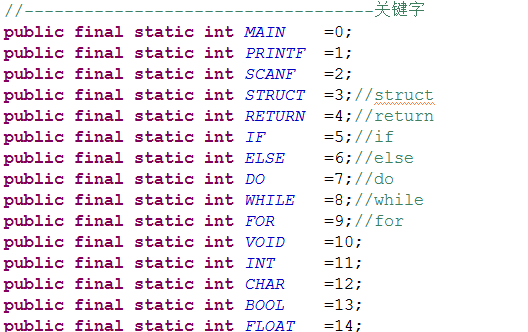


表3-1关键字定义表

界符表定义如下:

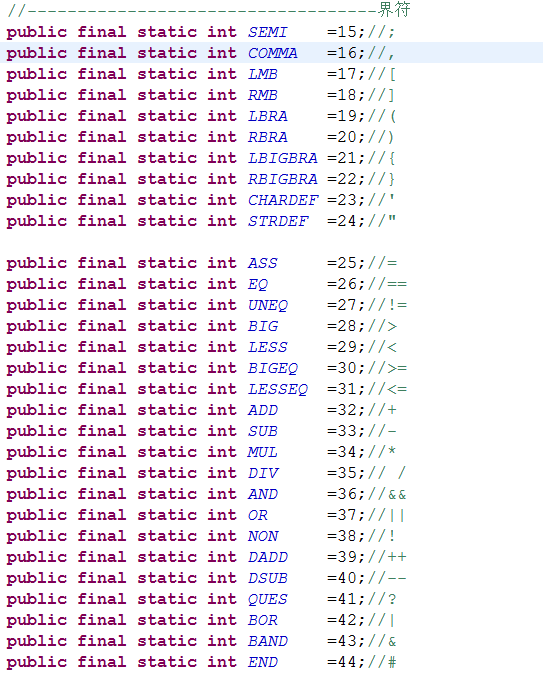


表3-2界符表定义

3.2.3算法

词法分析主要是通过自动机来写的，设计如下：

一个简单识别器（有限自动机）的设计，如图3-2-3-2所示:

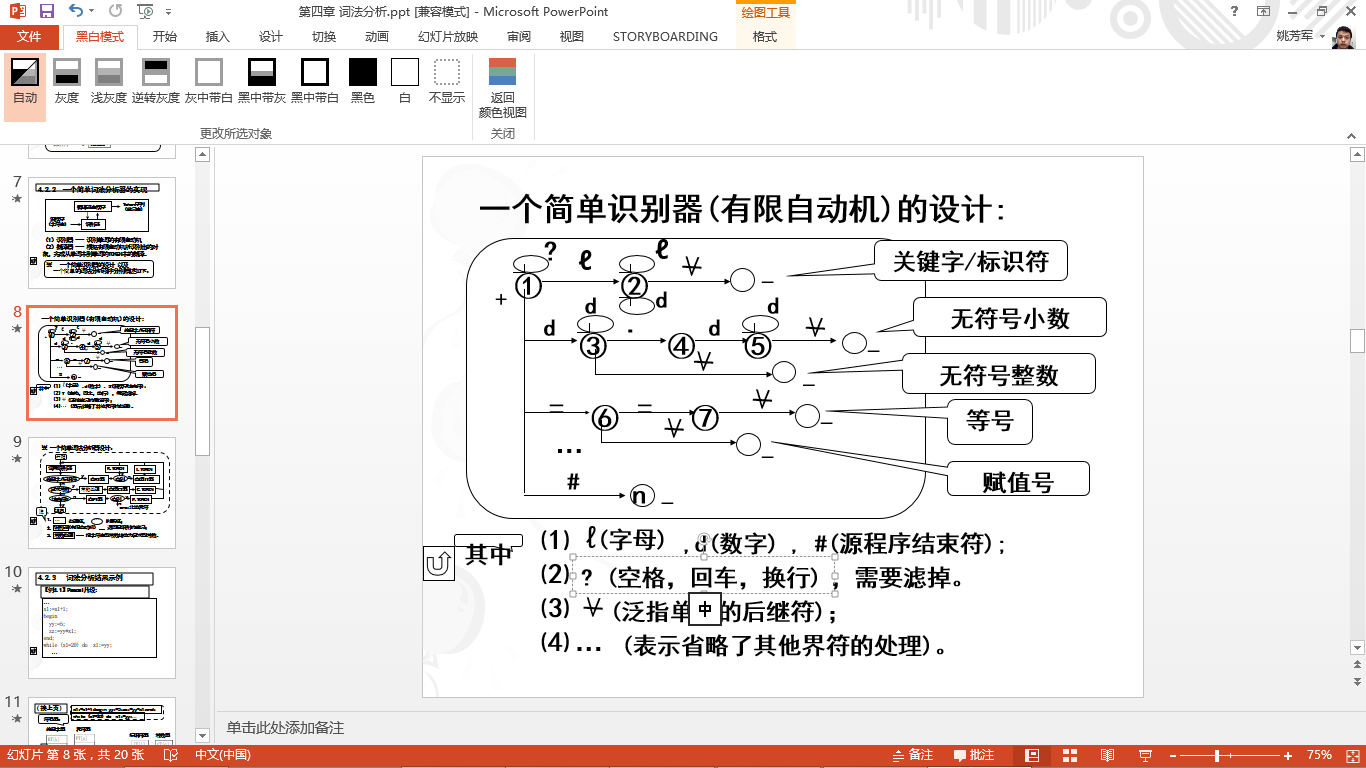


图3-7

<字母> 🡪 A|B|C|…|Z|a|b|c|…|z

<数字> 🡪 0|1|2|3|4|5|6|7|8|9

其中 (1)（字母），d（数字），#（源程序结束符）；

(2)？（空格，回车，换行），需要过滤掉；

　　 (３)（泛指单词的后继符）；

　　 (４) …..(表示省略了其他界符的处理)。

一个简单词法分析器设计，如图3-2-3-2所示：

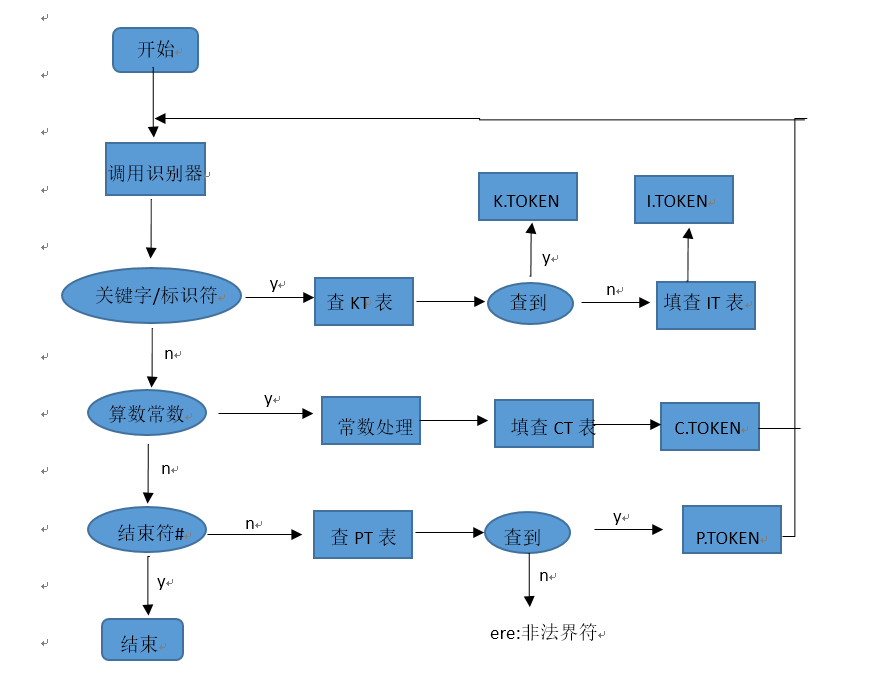


图 3-8

**3.3LL1分析法设计语法**

3.3.1 功能

语法扫描器的功能主要有三：

1. 识别一般源程序——检查源程序中是否符合语法规范；

（2） 实现控制语句的识别以及功能函数等的识别。

（3） 程序的纠错----对于源程序中出现的语法规范错误进行纠错改正并且提示。

3.3.2 数据结构

由于Parser类数据结构较为复杂，这里只简单列出成员变量

Parser类结构设计:



图3-9 Parser类结构设计

其中，除了用到了Error类进行错误信息的详细输出以及词法分析阶段用到的Word类和Token类等之外，因为涉及到语义分析阶段还引入了分析结点类(语义分析时再细说)以及符号表类中的总符号表(标识符结点类)以及函数表(函数结点类)、数组表(数组结点类)、结构体表(结构体结点类)、类型表(类型结点类)等。

AnalyseNode分析栈节点类结构



图3-10 AnalyseNode分析栈节点类结构

SynbolNode数据结构

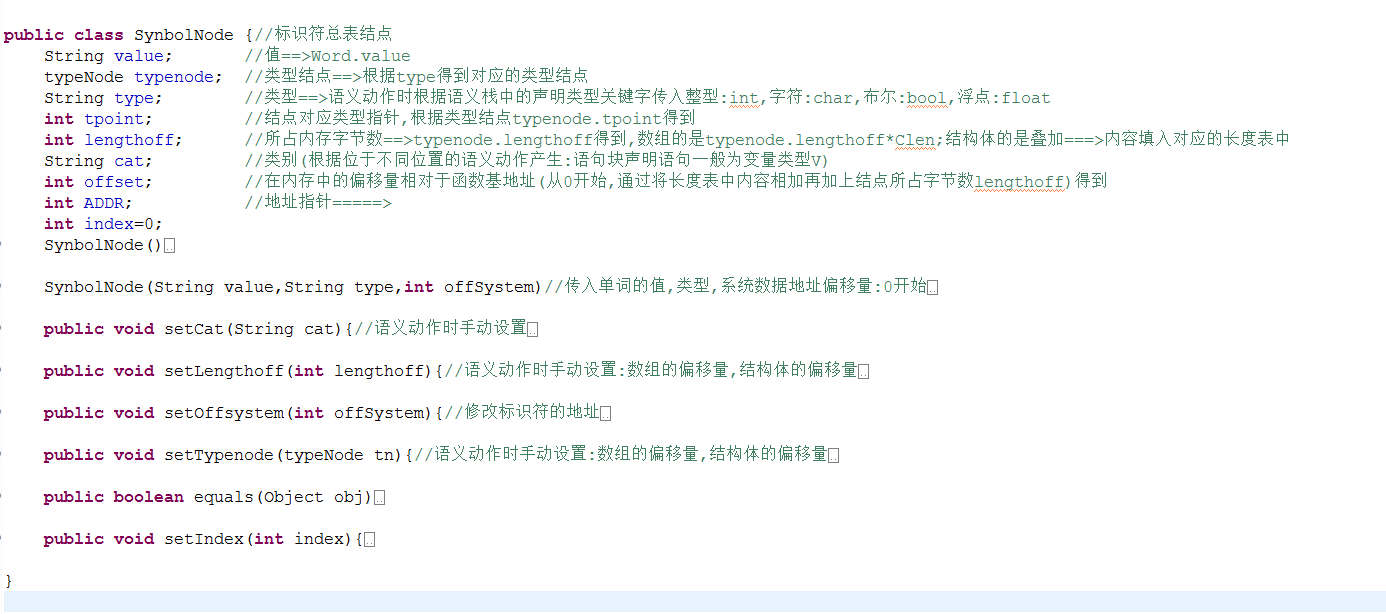


图3-11 SynbolNode数据结构

3.3.3程序生成文法：

这里就不给出在求证是否符合LL1文法属性时所求出的select集合了，经过验证确实符合文法属性。

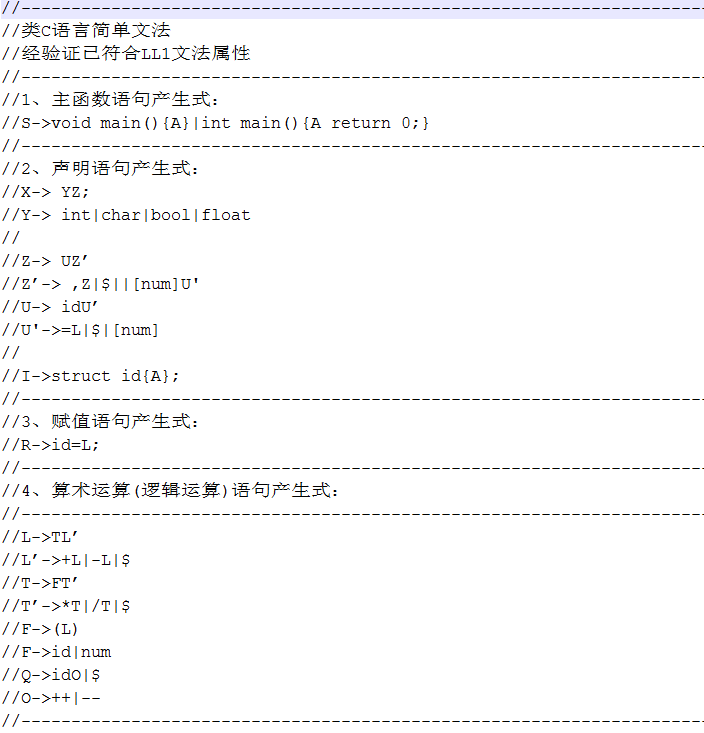


图3-12文法结构

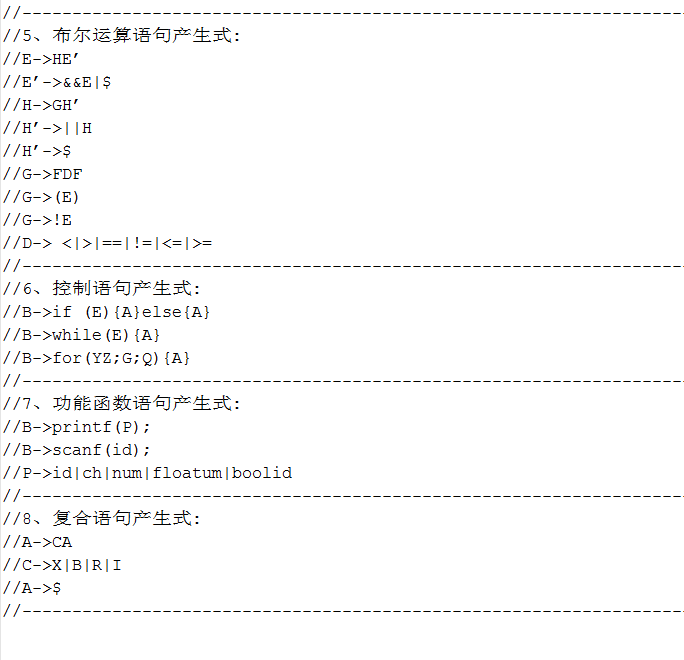


图3-13文法结构

3.3.4算法流程图：

LL(1)分析法是指从左到右扫描、最左推导(LL)和只查看一个当前符号（括号中的 1）之意；LL(1)分析法又称预测分析法，与递归子程序法同属于自顶向下确定性语法分析方法;

LL(1) 分析法的基本要点有三：

⑴ 利用一个分析表，登记如何选择产生式的知识；

⑵ 利用一个分析栈，记录分析过程；

⑶ 此分析法要求文法必须是 LL(1)文法。

这次实验我们用的就是LL(1)分析法，在程序设计开始前需要求出每个产生式对应的select集(篇幅有限，这里不再赘述)，引入栈结构，记录分析过程。一定程度上增加了我们的难度，但我们最终还是实现了。

**3.4 语义分析四元式以及符号表分析模块**

3.4.1功能

语义分析阶段的功能主要有三：

（1） 通过语法制导在文法中插入语义动作生成中间代码

（2） 实现生成中间代码以及符号总表

（3） 程序的纠错----对于源程序中出现的语法规范错误进行纠错改正并且提示。

符号表是标识符的动态语义词典，属于编译中语义分析的知识库，符号表可以存储标识符的各种信息，以便以后做处理。

3.4.2数据结构

由于语义分析是语法制导的，因此所用结构也为Parser类，所用的数据结构为栈(语法、语义栈)和链表(用于存储结点信息:符号结点等)。但其中加入了自定义数据结构:分析结点以及符号表类中的总符号表(标识符结点类)以及函数表(函数结点类)、数组表(数组结点类)、结构体表(结构体结点类)、类型表(类型结点类)等。还有用于生成四元式的四元式类。

Parser类结构设计:



图3-14 Parser类结构设计:

AnalyseNode分析栈节点类结构



图3-15 AnalyseNode分析栈节点类结构

SynbolNode数据结构

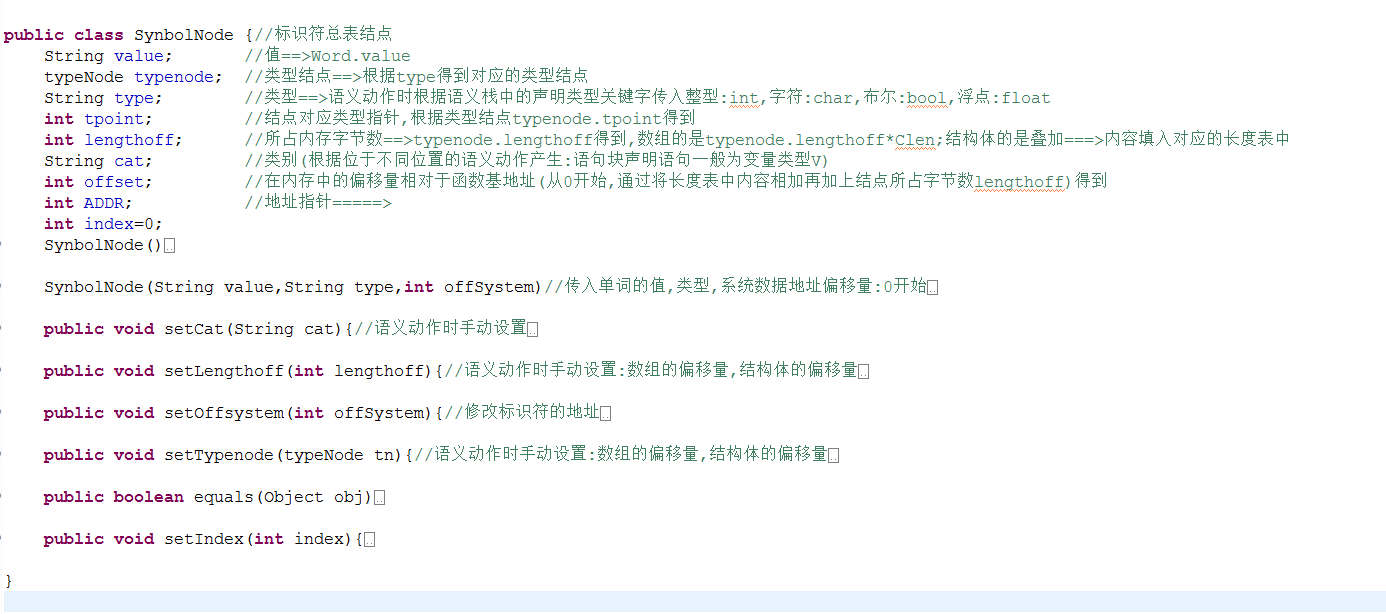


图3-16 SynbolNode数据结构

typeNode数据结构

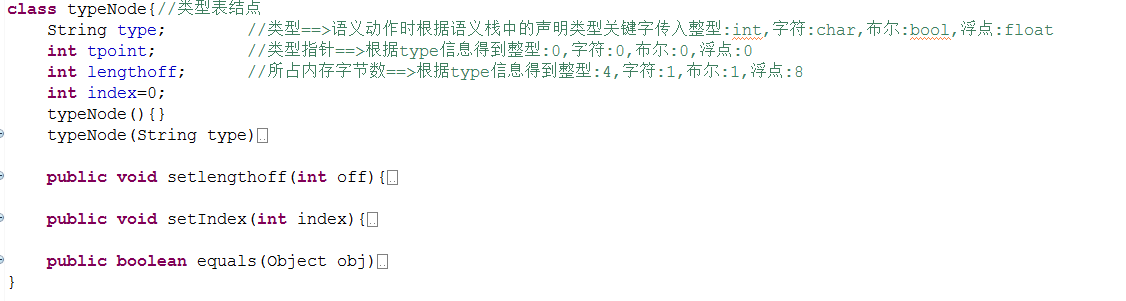


图3-17 typeNode数据结构

ainfNode数组结点数据结构

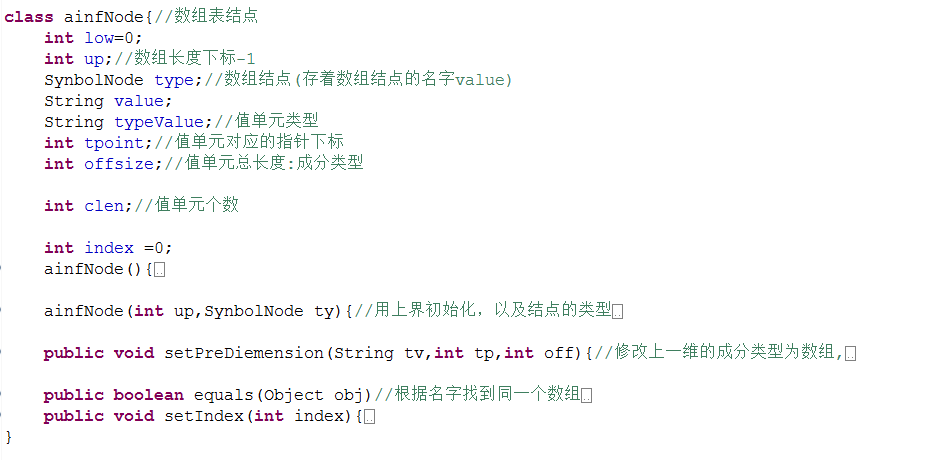


图3-19ainfNode数组结点数据结构

rinfNode数据结构



图3-20rinfNode数据结构

lengthNode数据结构

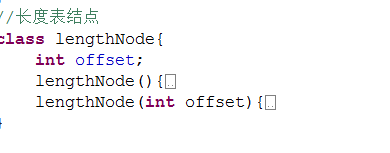


图3-21lengthNode数据结构

pfinfNode数据结构

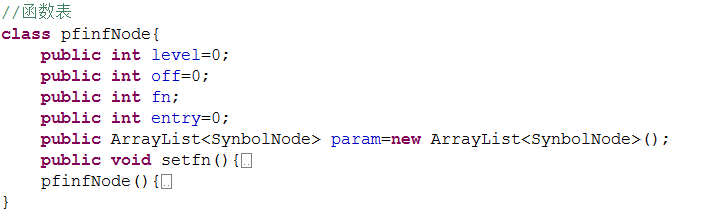


图3-22pfinfNode数据结构

FourElement数据结构

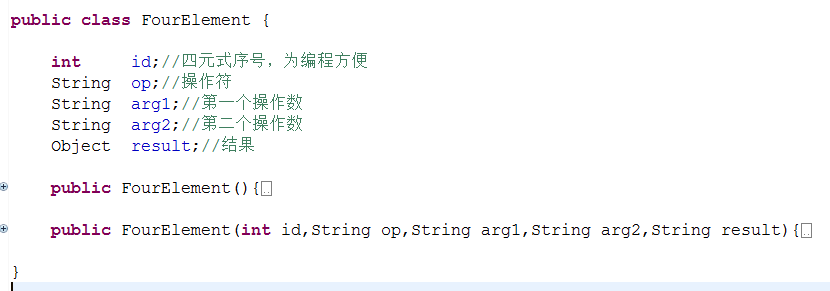


图3-23FourElement数据结构

3.4.3算法

对之前的文法进行语法制导翻译后的翻译文法如下:

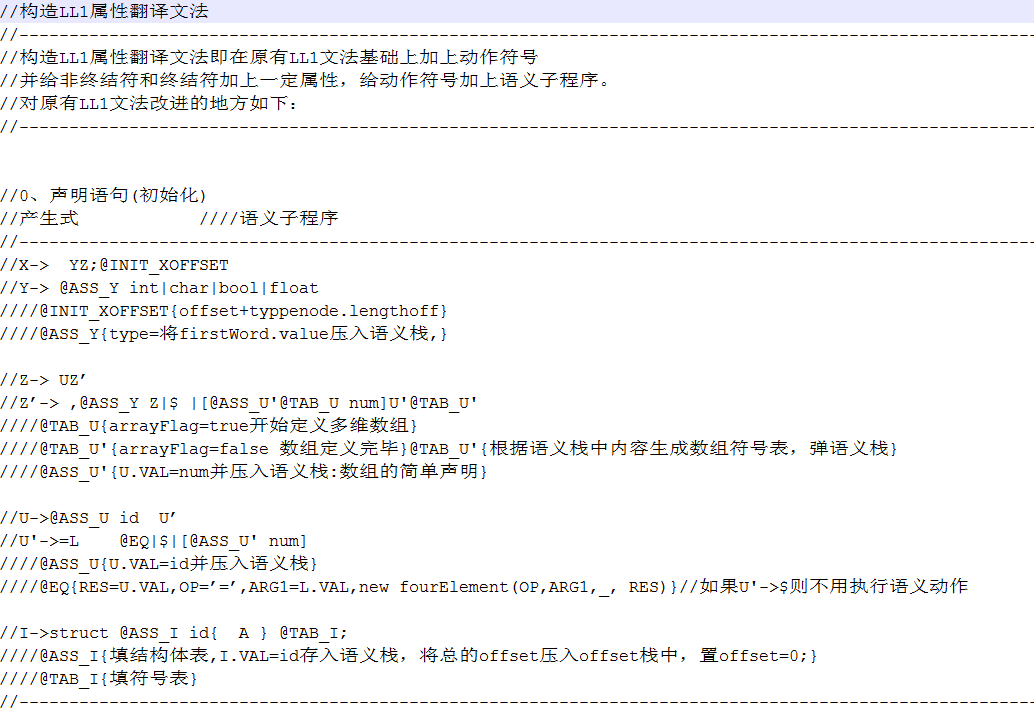


图3-24属性文法翻译

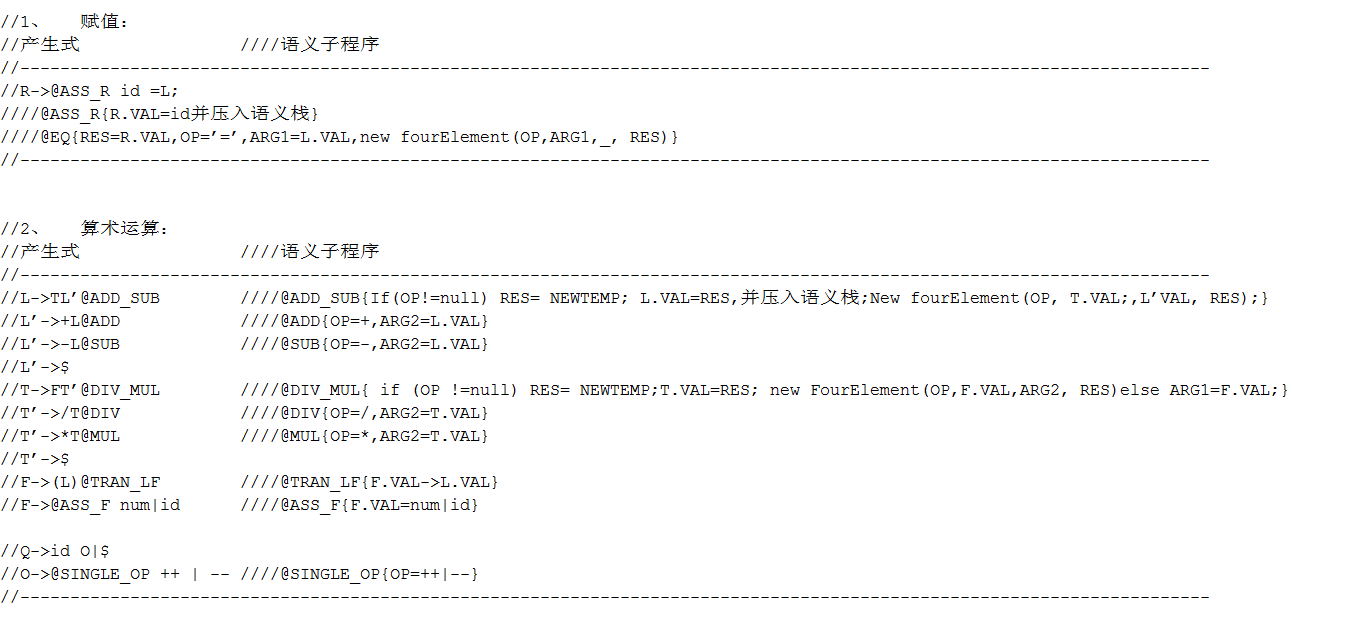


图3-25属性文法翻译

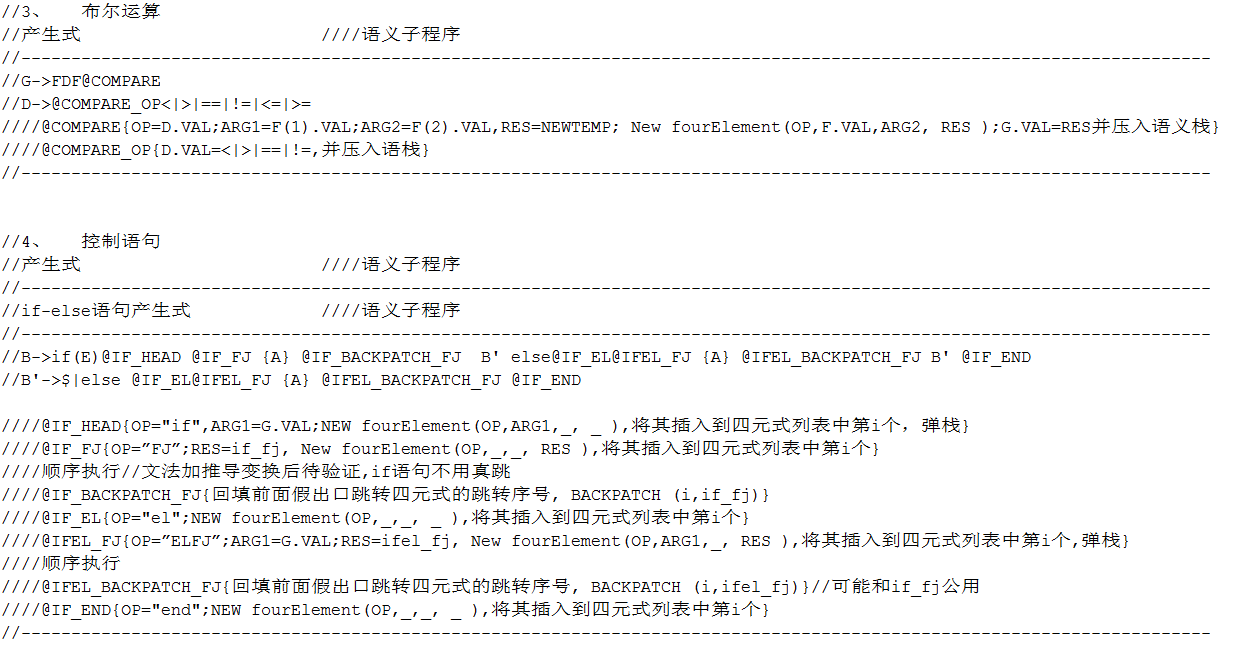


图3-26属性文法翻译

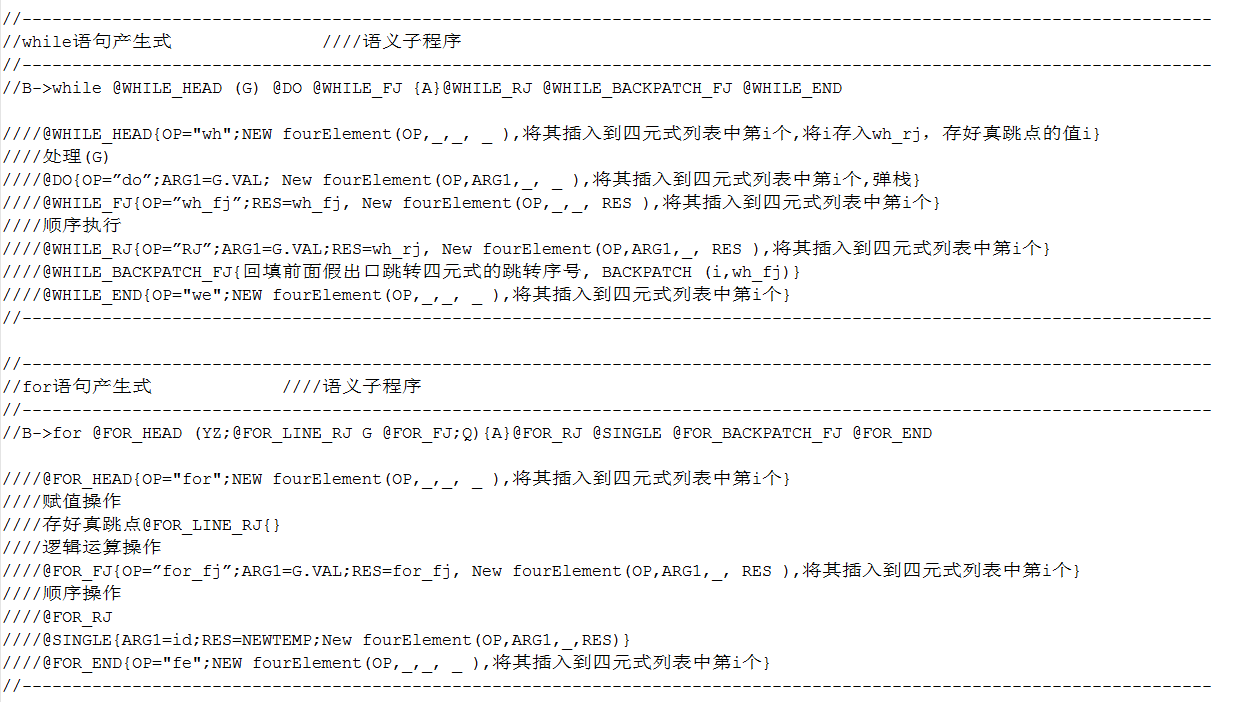


图3-27属性文法翻译

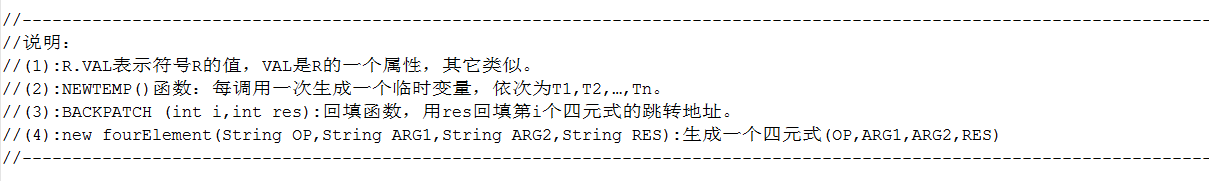


图3-28属性文法翻译

符号表的建立是在程序的声明部分，或者说是在函数的声明部分，因此我们只要对声明部分进行处理就可以了。

**3.5目标代码生成模块**

3.5.1功能

根据语义分析得到的四元式结合符号表生成对应目标机器的目标代码，由于水平限制我们只生成了简单的目标代码并未对其进行优化，不够完美。

3.5.2 数据结构

这里我设计了一个单独的类封装目标代码的生成，主要用的数据结构是链表，其中封装了之前所用自定义的四元式以及符号表类数据结构。

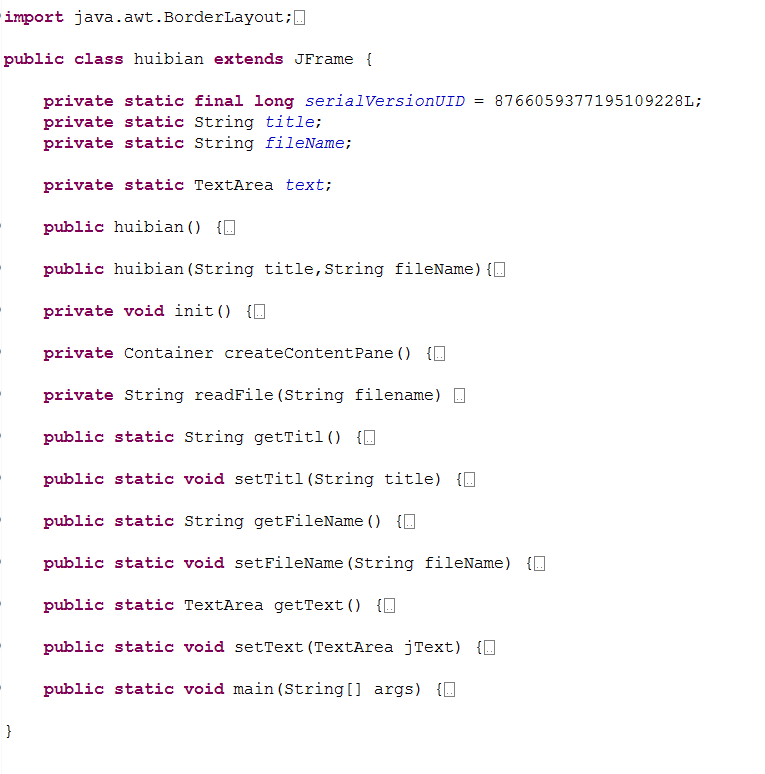


图3-29huibian类结构设计

3.5.3算法

利用LL(1)分析法语法制导翻译实现语义分析后，根据对应的目标机器生成对应的机器代码(目标代码)。它是指从左到右扫描、最左推导(LL)和只查看一个当前符号（括号中的 1）之意；LL(1)分析法又称预测分析法，与递归子程序法同属于自顶向下确定性语法分析方法;

LL(1) 分析法的基本要点有三：

⑴ 利用一个分析表，登记如何选择产生式的知识；

⑵ 利用一个分析栈，记录分析过程；

⑶ 此分析法要求文法必须是 LL(1)文法。

这次实验我们用的就是LL(1)分析法，在程序设计开始前需要求出每个产生式对应的select集(篇幅有限，这里不再赘述)，引入栈结构，记录分析过程。一定程度上增加了我们的难度，但我们最终还是实现了。

**4 程序设计与实现**

**4.1程序流程图**

4.1.1程序流程图

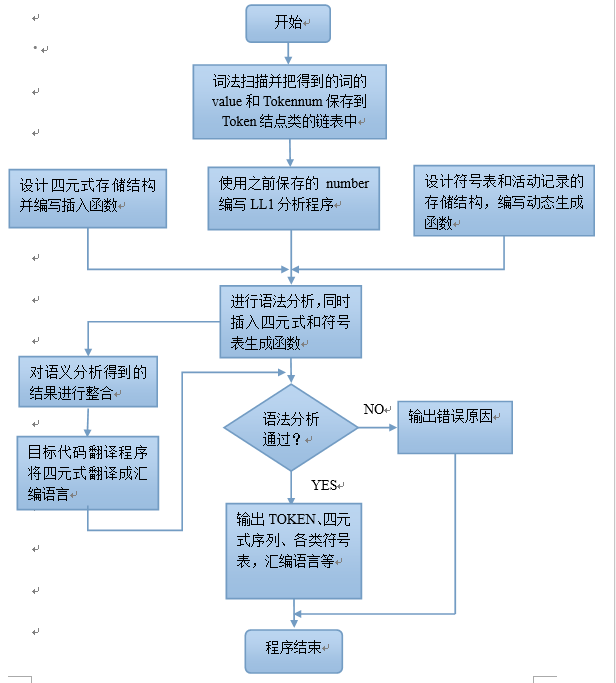


图4-0程序流程图

程序设计了可视化界面，按下不同的按钮可以解析输入(可以选择文本输入，也可以直接在文本输入窗口输入)进行输出不同的结果。

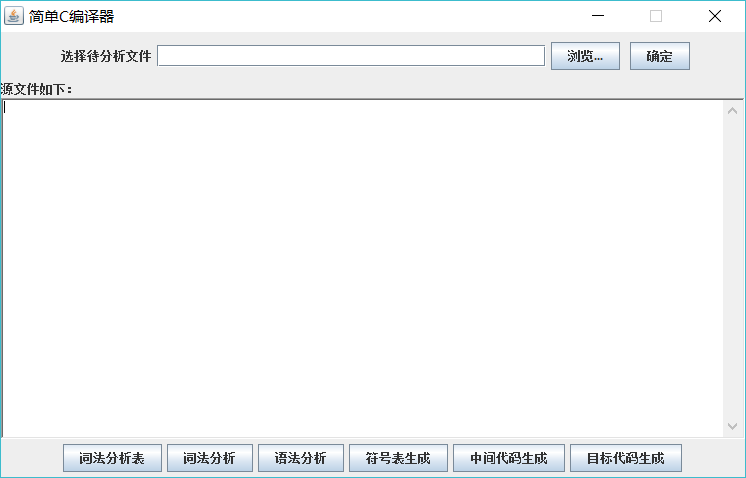


图4-1主界面图形界面截图

4.1.2设计框架

框架类图结构

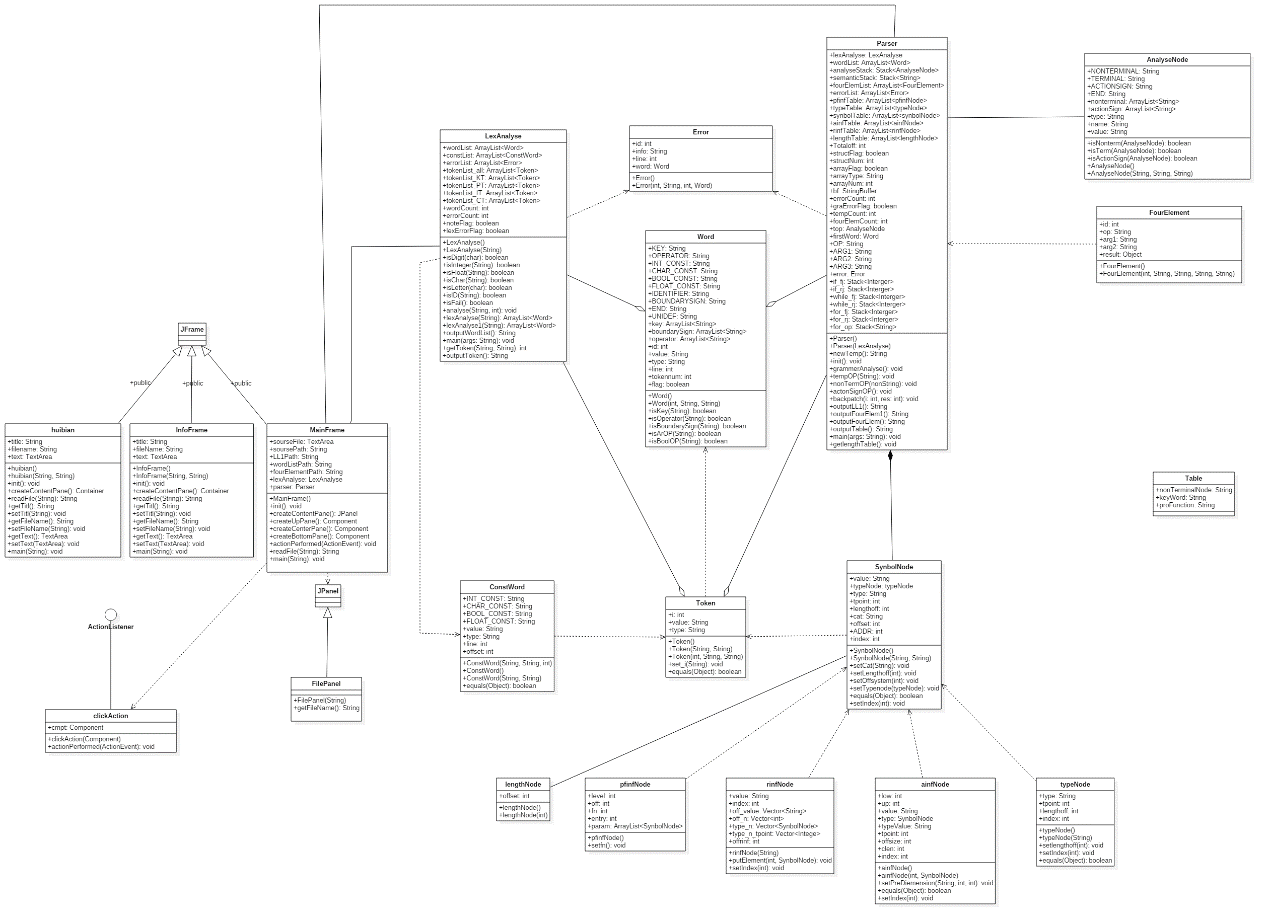
****

图4-2框架类图结构

4.1.3文法的设计

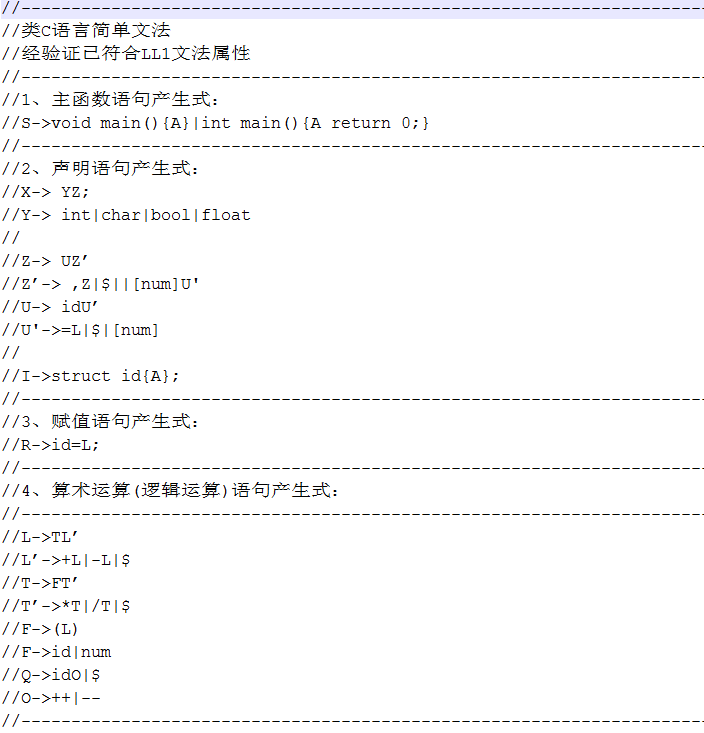


图4-3文法的设计

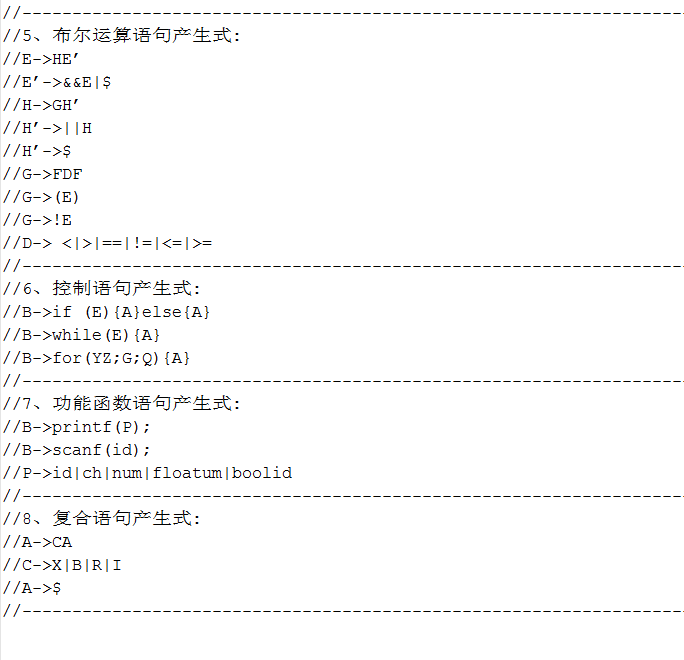


图4-4文法的设计

4.1.4翻译文法的设计

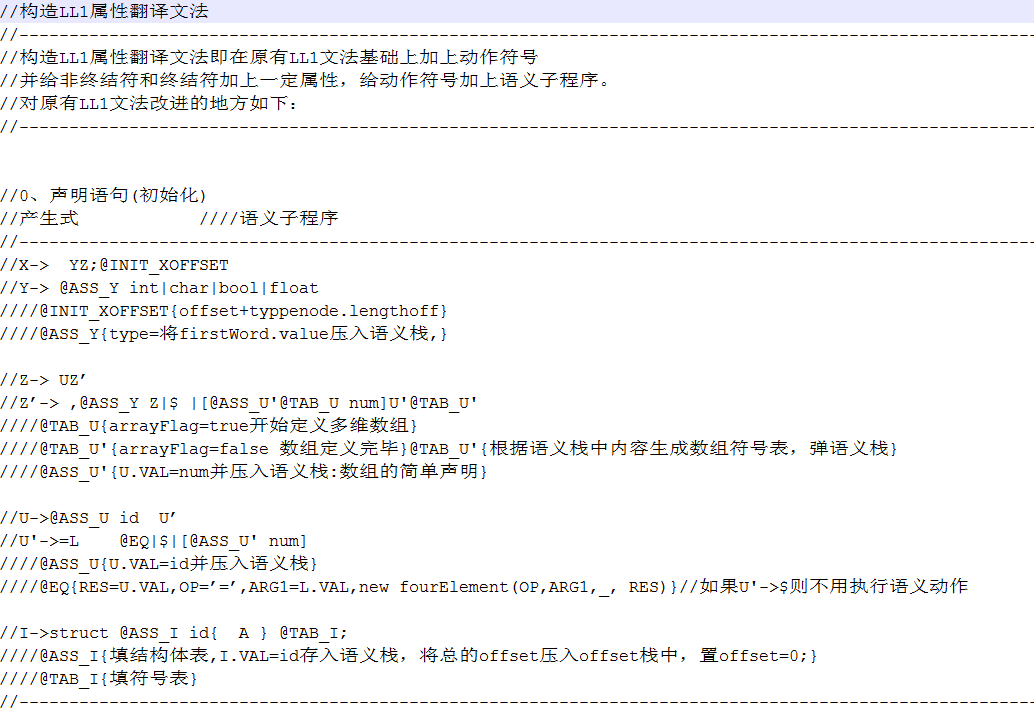


图4-5翻译文法的设计

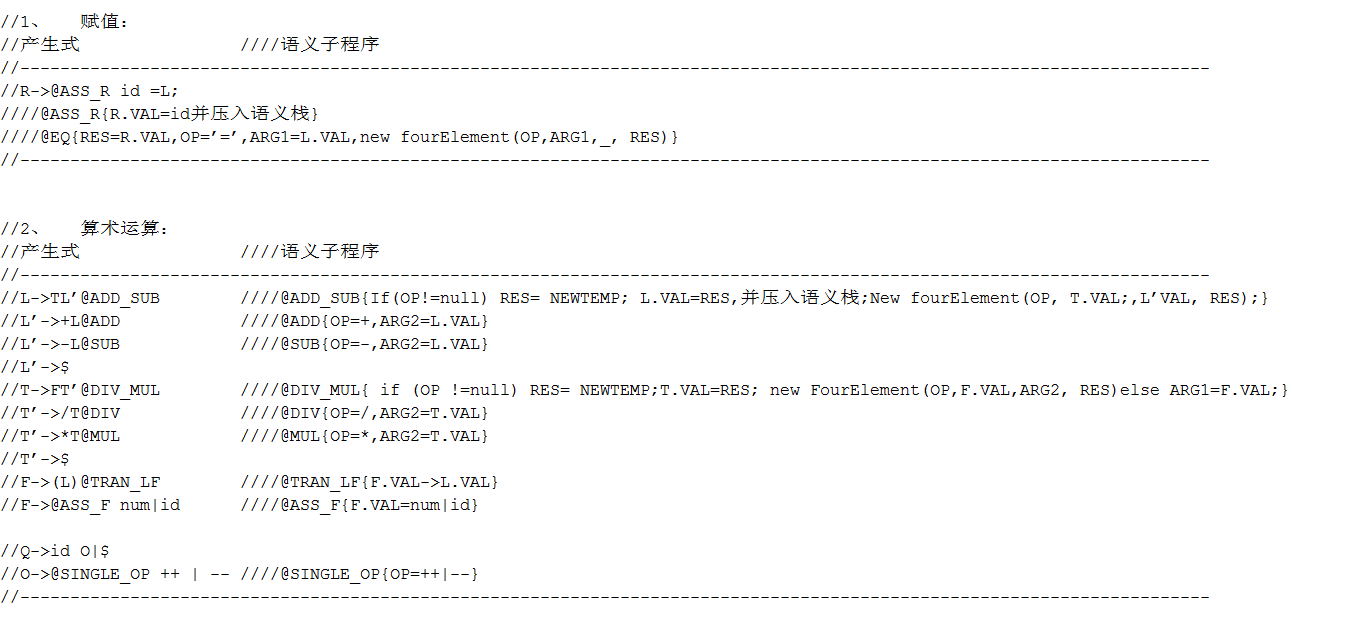


图4-6翻译文法的设计

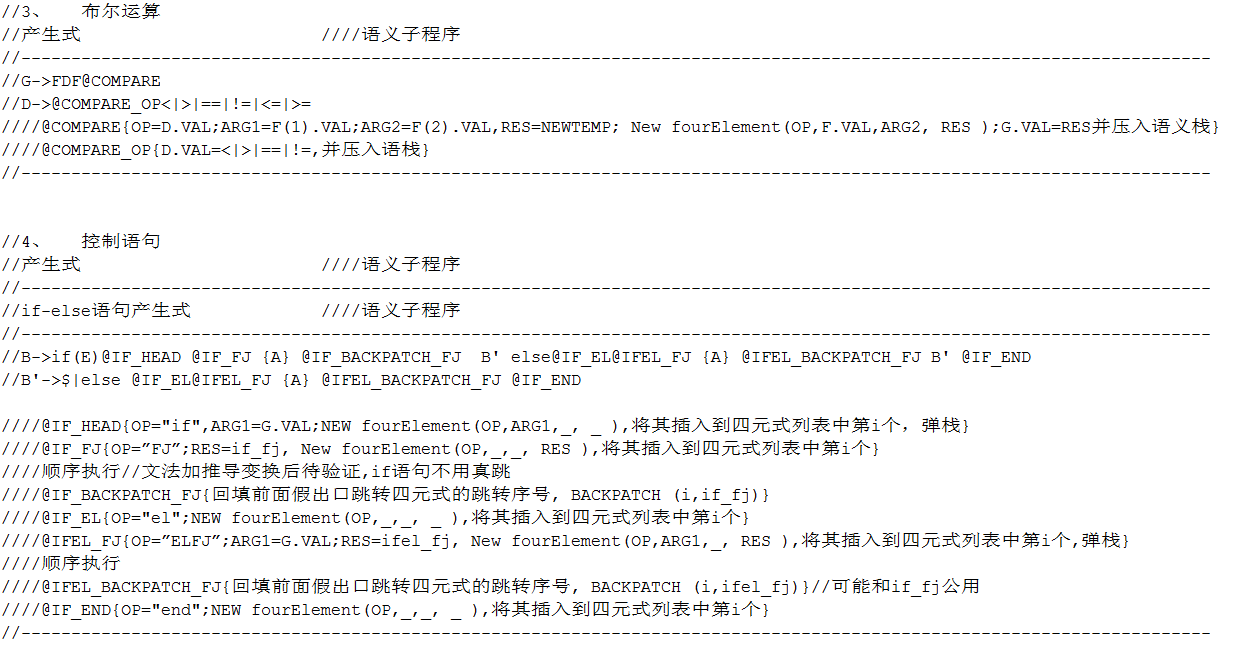


图4-7翻译文法的设计

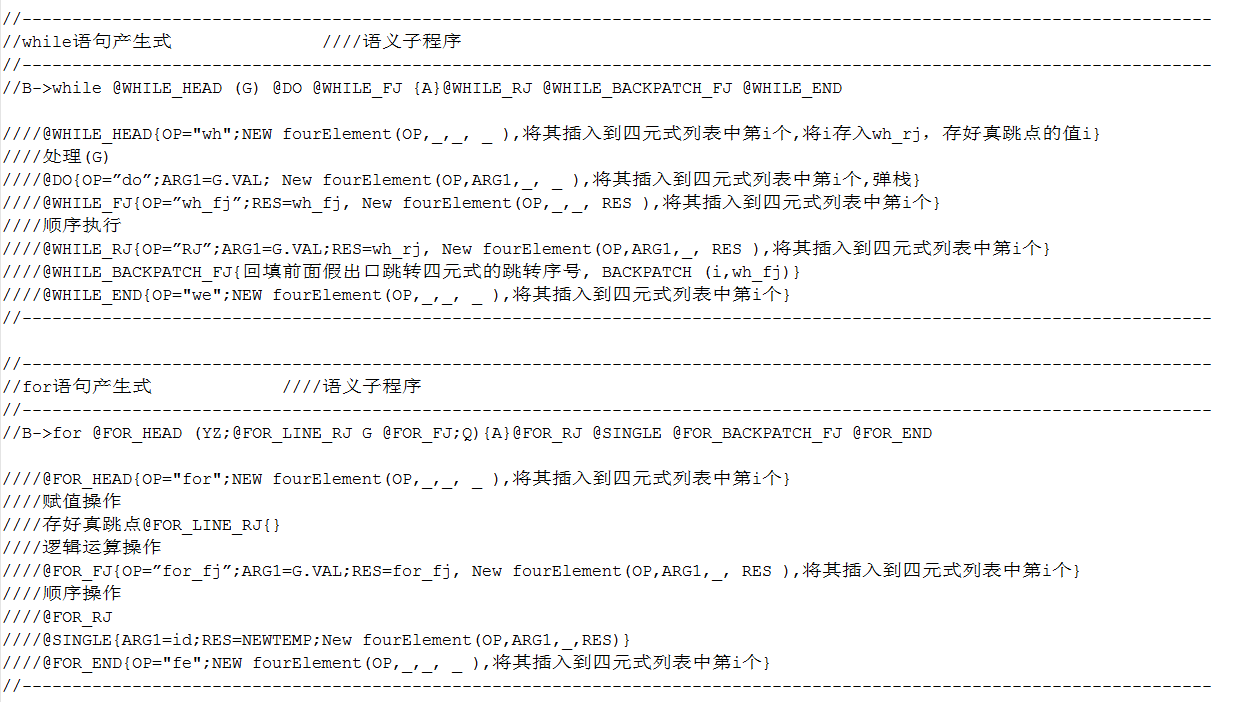


图4-8翻译文法的设计

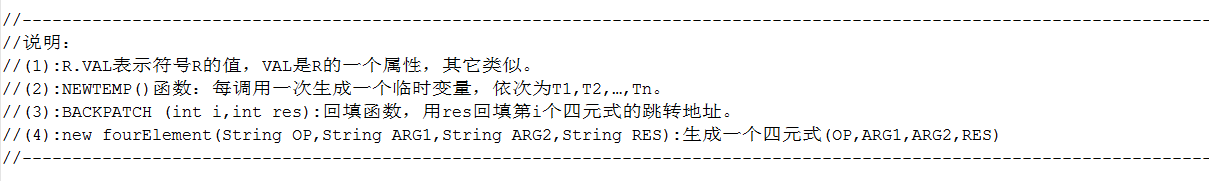


图4-9翻译文法的设计

**4.2程序说明**

4.2.1程序说明

说明：

1. 按照功能，分不同功能，分为2个package,分别为com.compiler(编译器包)、com.gui(显示ui包)
2. 每个package中可能包含若干个Class,实现具体的功能,组织结构如图4-2-1所示。

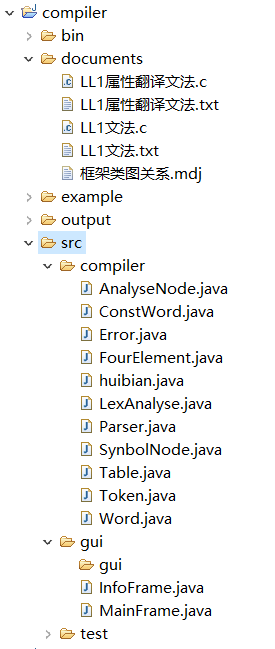


图 4-10

import包省去，仅显示package部分，以显示组织结构

4.2.2程序源代码

由于篇幅受限，这里暂时不展示出来。

**4.3 实验结果**

4.3.1测试用例

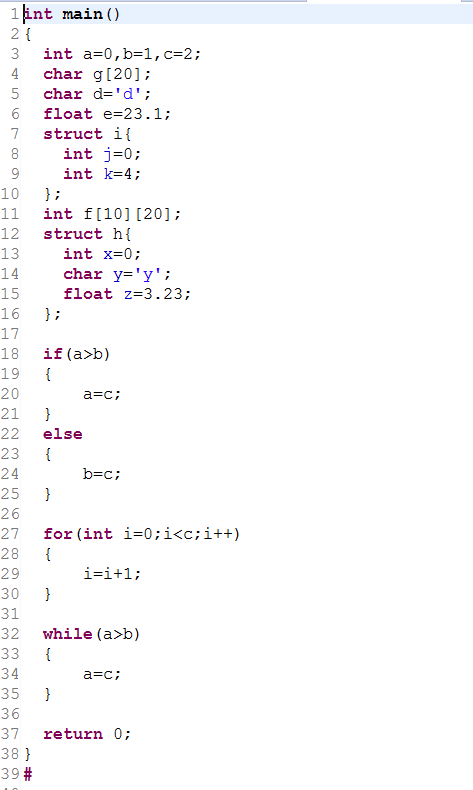


图4-11测试用例

4.3.2用例语法正确时结果

主界面：

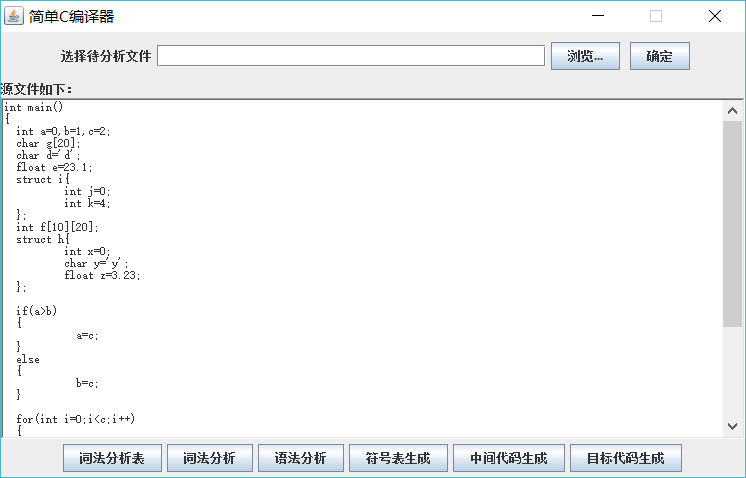


图4-12主界面

词法分析表：

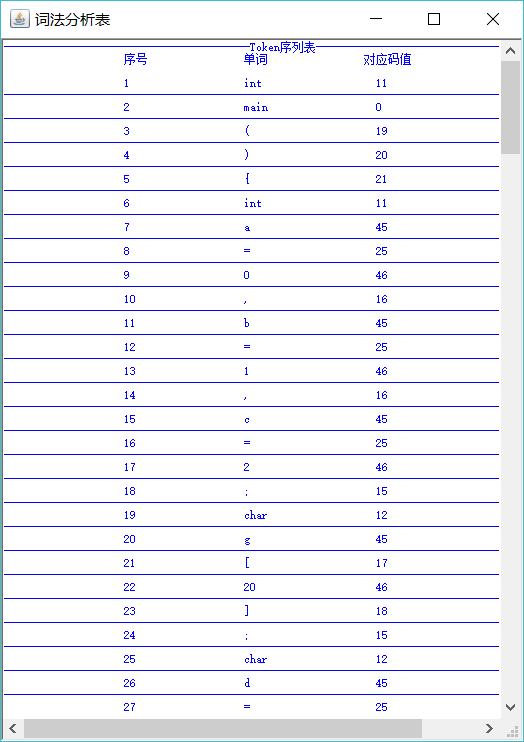


图4-13Token

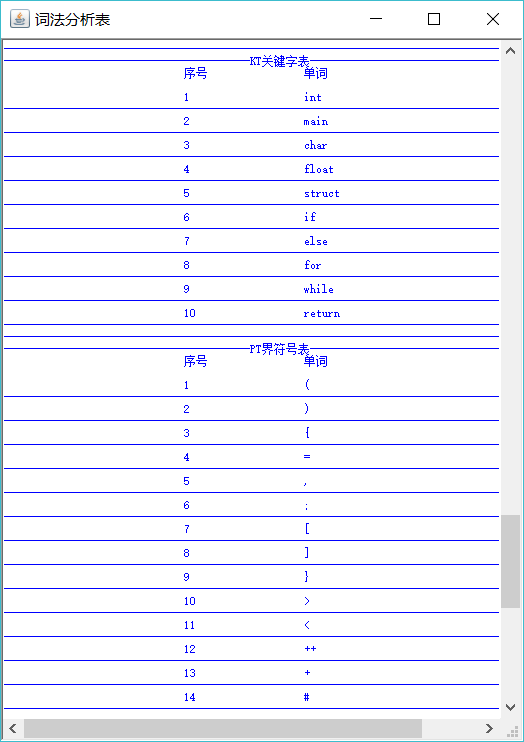


图4-14Token

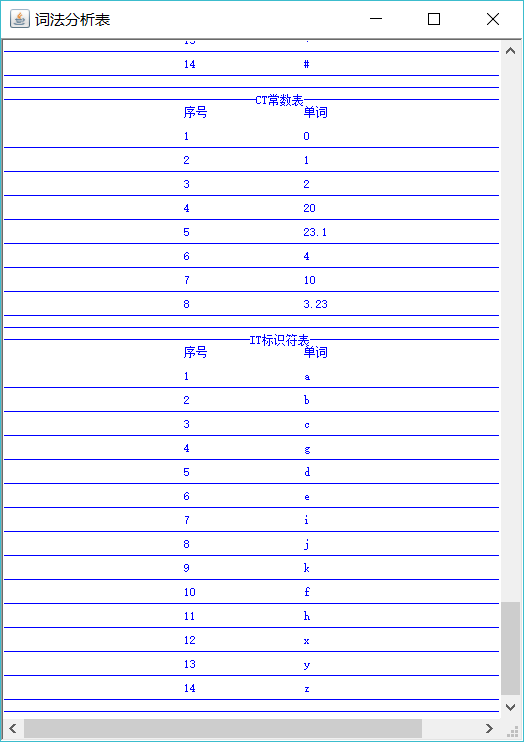


图4-15Token

词法分析：

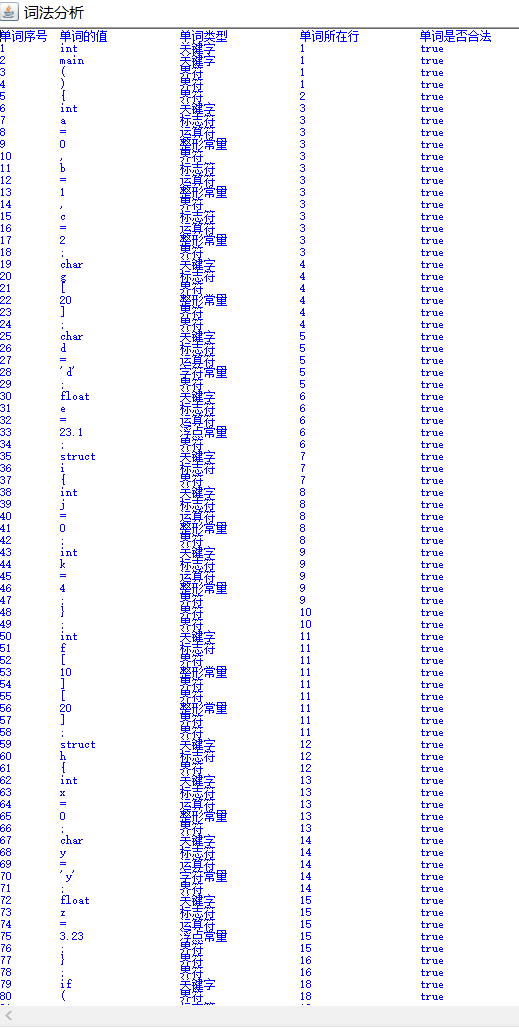


图4-16 测试截图

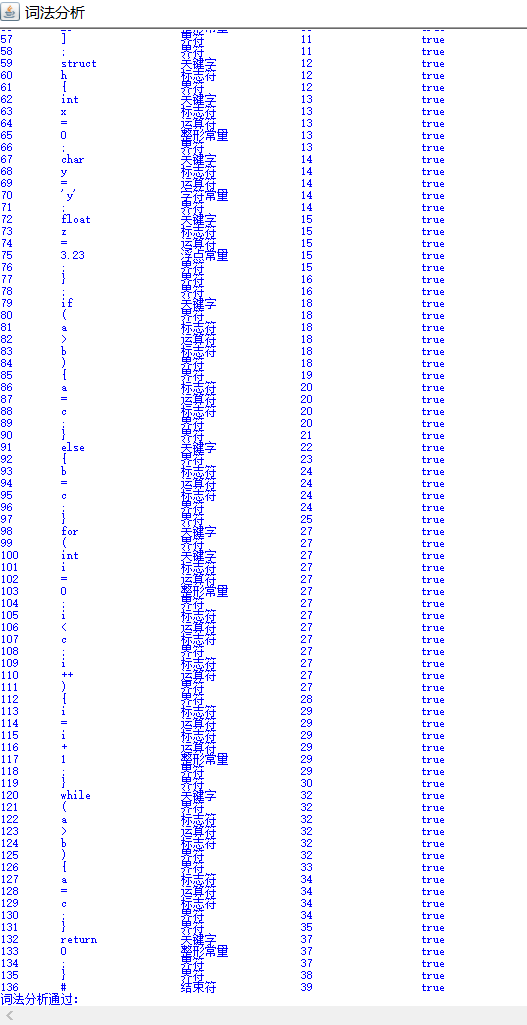


图4-17 测试截图

语法分析过程截图

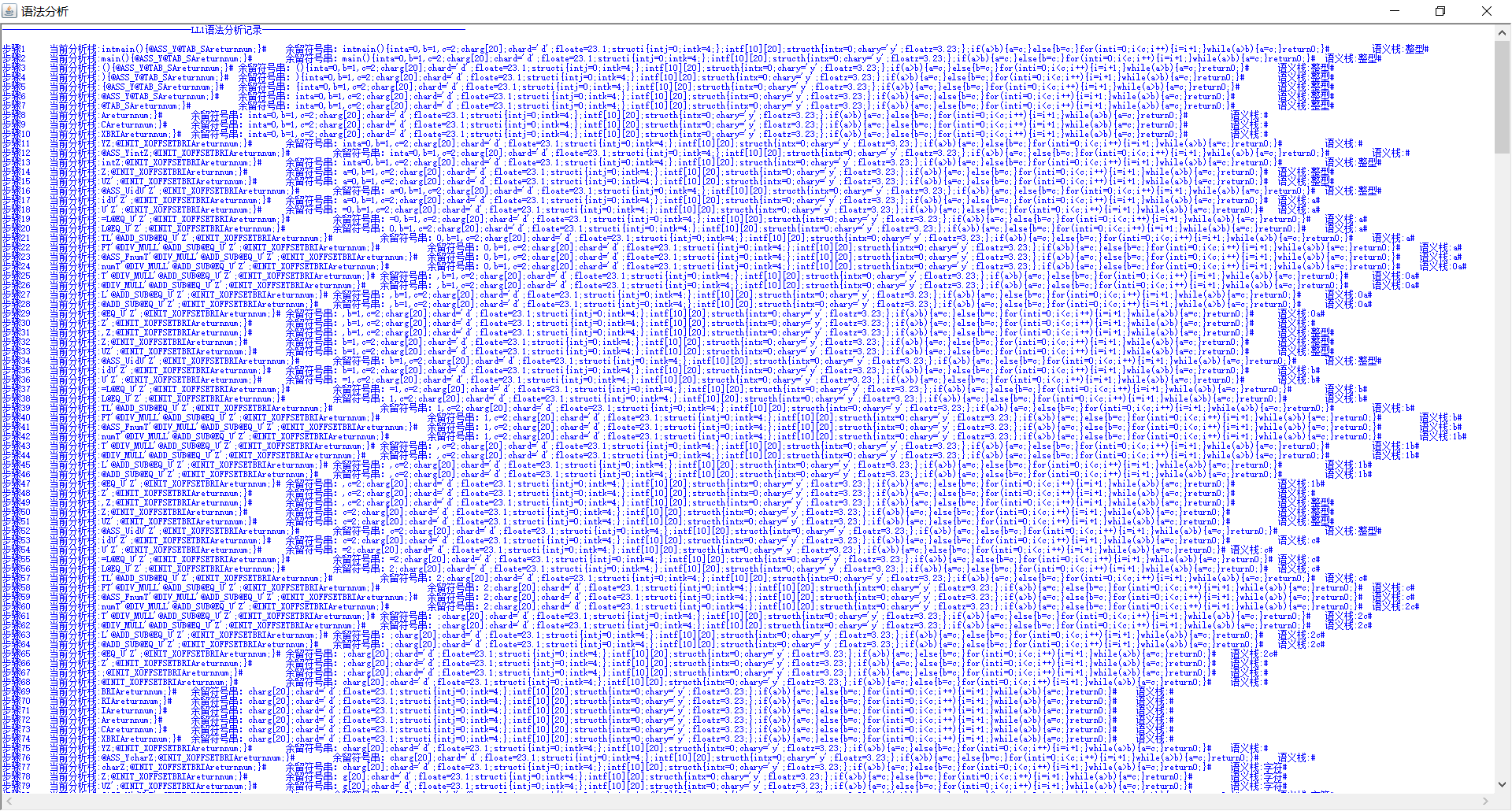


图4-18 测试截图



图4-19 测试截图

符号表输出：

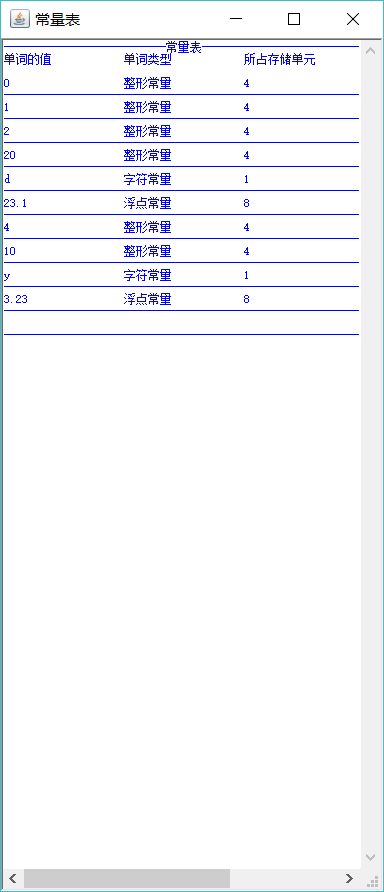


图4-20 测试截图



图4-21 测试截图

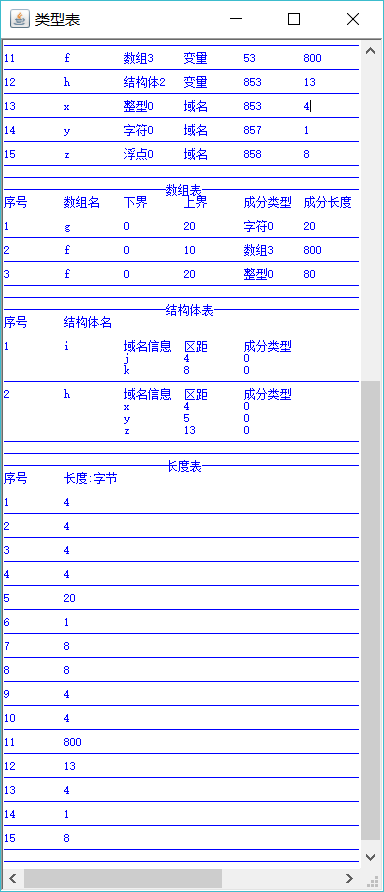


图4-22 测试截图

中间代码：

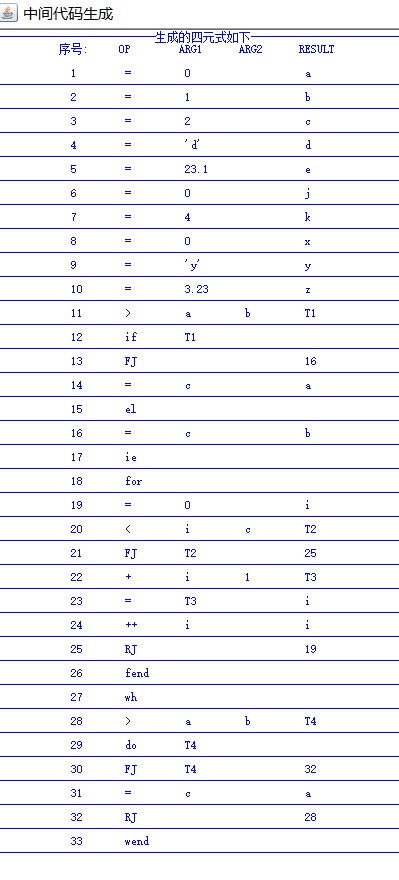


图4-23 测试截图

目标代码：

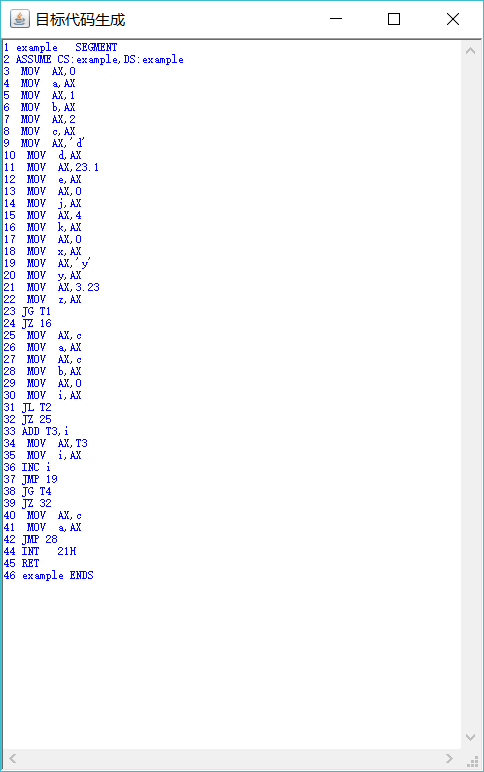


图4-24 测试截图

4.3.3用例语法错误时结果

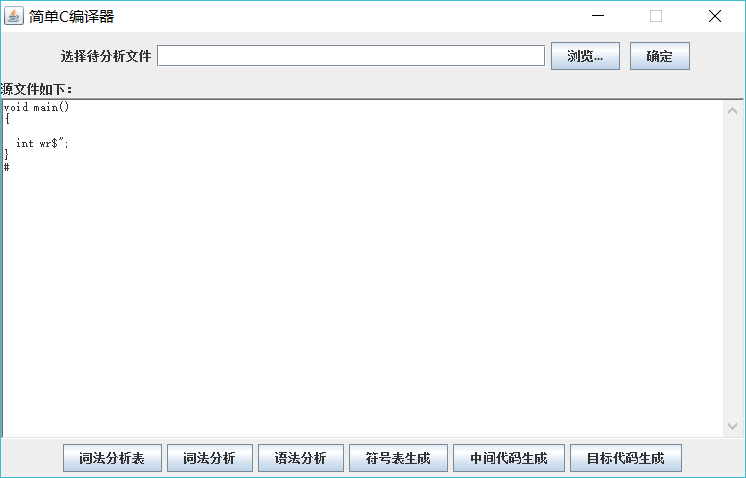


图4-25 测试截图



图4-26 测试截图



图4-27 测试截图

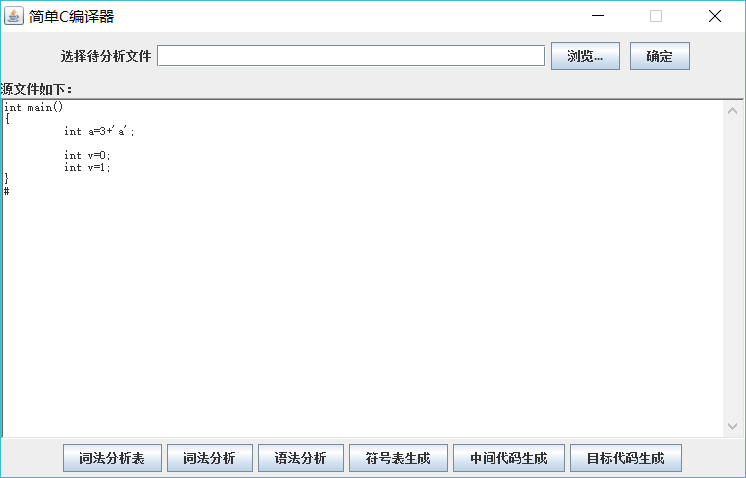


图4-28 测试截图

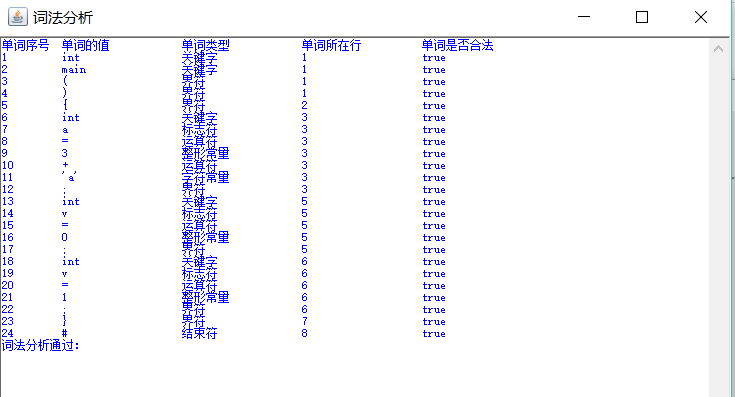


图4-29 测试截图



图4-30 测试截图

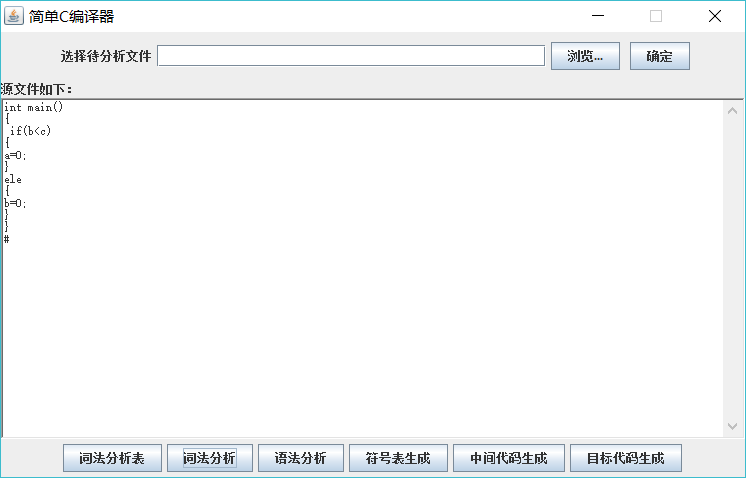


图4-31 测试截图

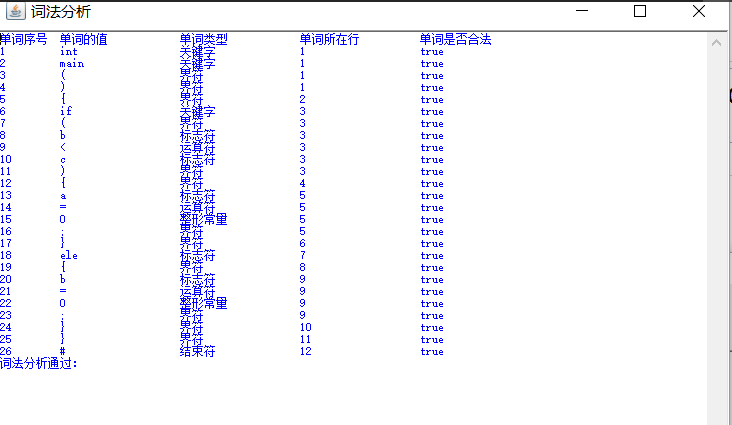


图4-32 测试截图

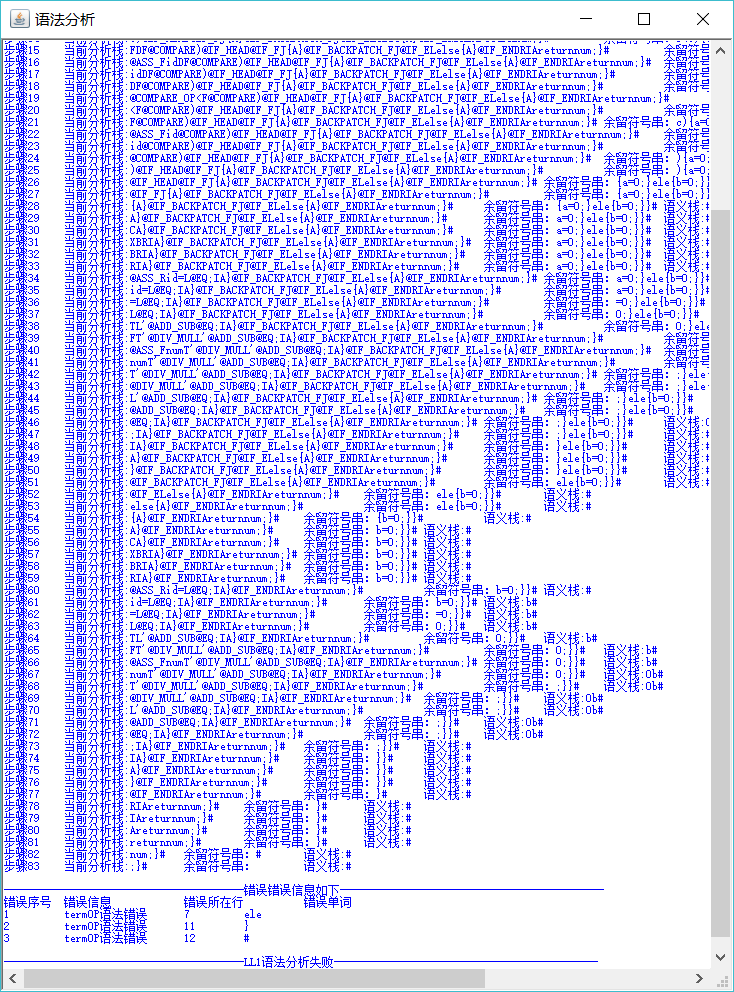


图4-33 测试截图

**5.结论**

本次实验我们采用了Java语言来编写一个类C语言的编译器，步骤按照先自动机词法扫描源程序同时生成Token序列，然后运用LL1分析方法进行语法分析，接着往里面插语义动作语法制导同步生成四元式与符号表构建。最后是对语义分析产生的中间代码结合符号表生成目标代码。

词法扫描是通过读文件模式，将代码和关键字表与界符表从文件读入Java的数据结构HashMap,采用键值对的形式存储。之后就是运用Java API中独有的String类的各类函数对代码进行有限自动机处理，最终生成了Token。

语法分析通过自建文法，采用LL1分析法，求出所有产生式的select集合然后根据select集合进行语法分析，然后一步一步实现在此之中生成了四元式，其中表达式的四元式是采用逆波兰式的方法生成的，于此同时完善符号表。

目标代码的生成是由四元式结合了符号表生成的，这个必须指定目标机器才有意义，我们这次目标代码的指定目标机器是X8086。

其中的特色点有：各分析阶段能够检测出错误，并且能指出错误在哪一行，具体为什么错误；表示式的四元式采用了逆波兰式的方法；同时控制语句, 我们的编译器能判断其中的boolean表达式的真值，从而能采用正确的逻辑得出正确的结果；符号表全面完善；目标代码符合目标机器要求。

**6.参考文献**

1、陈火旺.*《程序设计语言编译原理》（第3版）*. 北京：国防工业出版社.2000.

2、美 Alfred V.Aho Ravi Sethi Jeffrey D. Ullman著.李建中，姜守旭译.《*编译原理*》.北京：机械工业出版社.2003.

3、美 Kenneth C.Louden著.冯博琴等译.《*编译原理及实践*》.北京：机械工业出版社.2002.

4、金成植著.《*编译程序构造原理和实现技术*》. 北京：高等教育出版社. 2002.

**7.收获、体会、建议**

**7.1收获、体会**

本次《编译原理》课程设计我们都学习到了许多，一方面是真正把上课所学的各章理论知识联系起来，完成了一个完整的编译器的设计制作；另一方面我们通过课程设计也提升了自己的自信心。从一开始拿到课题担心最终完不成，到最后我们不仅仅制作了编译器前端，这个过程中我想我们都收获了许多。当然，就像农民伯伯秋天的收获需要大半年的辛勤一样，我们最终目标的达到并不是一蹴而就的，需要在成功前不断地努力。我们所有组员为这个目标的达成付出了许多，尤其是作为组长，两周的时间几乎天天都在写代码和调试互相之间的接口，甚至于需要不断地黑盒测试来发现其他模块的错误。虽然过程很辛苦，但是能够为整个小组最后目标的达成，这种奉献精神是必要的。从这个角度来说，《编译原理》课设不仅仅是一次写代码的过程，也是一次发现自我，增强团结协作意识的机会。在搭建框架的时候使用了UML语言进行程序设计，还使用了Github的版本控制功能，能够使我能四个人的项目随时同步。当然了，在编译原理知识我也花了不少心思，比如词法分析，语法分析，四元式，符号表的内容我都了解了，在队友们编写程序的时候我也经常一起讨论知识点，这样就进一步巩固了我的知识点。总之这次实验收获还是挺多的。

**7.2建议**

课设时间提前是非常明智的选择，既给了同学们提前复习的机会，也减轻了最后期末阶段的压力。但是有一点是既然是课程设计，希望老师应该多给我们项目经验，如何开展一个项目，中间得注意些什么，毕竟大家大多数还是得进公司发展，比较贴近实际需求。同时也希望计算机学院方面重视同学们的实战经历，说实话，没有第一次的实战经历，同学们压根没有热情、没有兴趣开展编程、更谈不上学理论知识了。但是若是开展了第一次的实战项目，以后同学们不管学习什么科目，都会有兴趣学习理论知识，从而达到理论与实践相结合的目的。最后感谢老师的大力帮助。还有辛勤的指导。