合肥工業大學

编译原理课程设计报告

设计题目	将 FOR 语句转换成四元式的程序实现
学生姓名	周布伟
学　　号	2019217192
专业班级	计算机科学与技术 19-2 班
指导教师	李宏芒老师、唐益明老师
完成日期	2021 年 7 月 4 日

目　录

一、 设计目的及设计要求

设计内容：设计一个语法制导翻译器，将 FOR 语句翻译成四元式。

要求：先确定一个定义 FOR 语句的文法，为其设计一个语法分析程序，为每条产生式配备一个语义子程序，按照一遍扫描的语法制导翻译方法，实现翻译程序。对用户输入的任意一个正确的 FOR 语句，程序将其转换成四元式输出(可按一定格式输出到指定文件中)。

二、 开发环境描述

型号：HUAWEI MateBook 13

处理器：Intel(R)Core(TM) i7-8565U CPU @1.80GHz 1.99 GHz

已安装的内存(RAM)：8.00GB

系统类型：Windows 10 家庭中文版，64 位操作系统，基于 x64 的处理器

硬盘名称：WDC PC SN720 SDAPNTW-512G-1027

采用软件：Visual Studio 2019

采用编程语言：C#

三、 设计内容与主要算法描述

(一) 问题描述与解决方法

本次课程设计根据编译的基本过程，设计实现语法制导翻译器，具体实现过程包括词法分析、语法分析、语义分析和中间代码产生。设计与实现过程也将围绕这几个步骤展开，但由于需要实现语法制导翻译，文法的设计与翻译模式的设计将同时进行，具体而言：

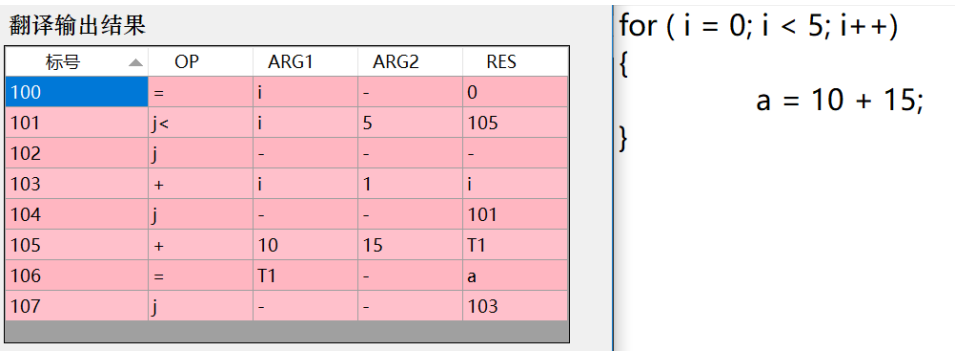
图 1 语法制导翻译器实现过程

1. 设计实现词法分析器。依托 C++语言的语法与命名习惯，识别输入文本中的 C++语言代码，判断输入符号类型，产生二元序列和位置坐标。同

时将识别后的代码文本进行处理（如去掉换行、空格、制表符，将 for 单词转化为 f 终结符，将常数、变量用相应终结符代替等）以作为语法分析的输入串，记录并保存代码文本中出现的常量和变量以作为翻译时部分终结符的输入属性。

2. 设计实现语法分析器。采用 LR(1)文法进行 FOR 循环语句的设计，同时考虑翻译模式。由于采用的是自底向上的翻译流程，在文法设计时需要插入非终结符以产生相应的语义动作，在 LR(1)的项目集族和预测分析表的实现时需要考虑产生式出现空字的情况。最终语法分析器的总控程序通过得到的预测分析表对输入串进行自底向上分析。

3. 设计实现语法制导翻译器（语义分析和中间代码生成器）。在设计文法的同时考虑 FOR 语句的翻译模式，其中包括赋值语句、条件控制语句的生成等，为每条产生式配备语义子程序，在翻译模式的基础上，结合语法分析器，当语法分析过程出现归约时执行相应的语义动作。最终翻译的结果以四元式的形式给出。



标号	OP	ARG1	ARG2	RES
100	=	i	-	0
101	j<	i	5	105
102	j	-	-	-
103	+	i	1	i
104	j	-	-	101
105	+	10	15	T1
106	=	T1	-	a
107	j	-	-	103

图 2 设计实现的部分成果预览

(二) 词法分析器的实现

1. 主要功能描述

本次课程设计的词法分析器能够统计行数和列数用于错误单词的定位删除空格类字符，包括回车、制表符空格；识别单词（关键字、标识符、常数、运算符、关系运算符、分界符号），并用（内码，属性）二元式表示；若发现错误则报告出错；填写标识符表供语法制导翻译过程使用；识别并跳过注释；判别不合法的浮点数和组合运算符。

2. 实现算法描述

本程序通过外部文件读入的方式进行分析，通过 C# StreamReader 对象实现

对文本的读入与回退，采用超前搜索，通过接收字母、数字、符号等信息，采取相应的处理措施，具体过程如**错误!未找到引用源。**所示。

具体而言，对于关键字、标识符、运算符等信息的识别，在识别的过程中用变量保存缓存字符串，输入字符、字符（串）所在的行列位置等信息，本程序采取的算法如下：

(1) 对于关键字的识别，当输入的字符为字母时，继续读入字母或数字，以其他符号结束时回退一格，并将拼接的字符串查找所存储的关键字表，查找成功则为关键字，否则就查找标识符表，若不存在则加入标识符表；

(2) 对于常数的识别，当输入的字符为数字时，继续读入数字，以其他符号结束时回退一格（后面的功能会考虑浮点数以及数字后接字母构成非法标识符的情况），然后将读入的数字（默认为十进制）转换成二进制的形式，并查找常数表，若不存在则加入其中；

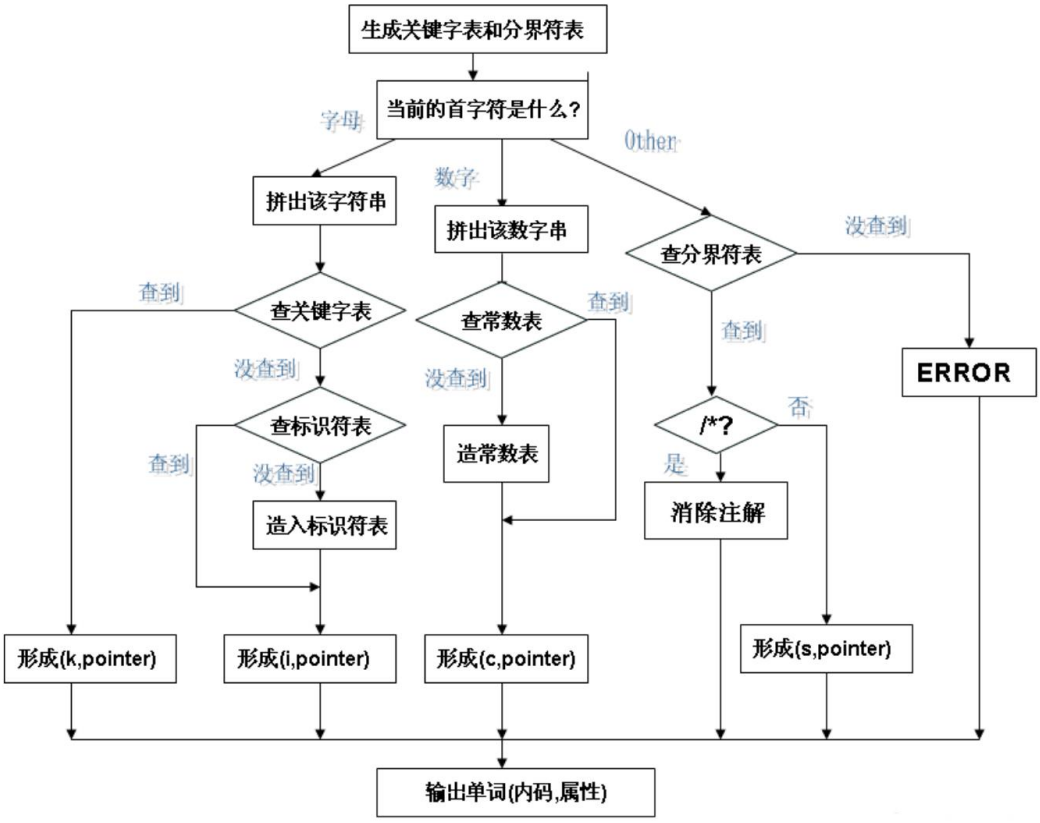


图 3 词法分析器实现过程（图：《编译原理实验指导书》）

(3) 对于各种符号的识别，为了避免出现读入过多符号而导致识别出错的情况，本程序将各类符号分成分界符、算数运算符、关系运算符，分别保存在 S1, S2, S3 中，并分开识别。当识别输入的字符为符号时（包括所

有符号），继续判断是否为分界符，若是则输出，不是则判断是否为算数运算符，是则输出，不是则继续读入新的字符，形成字符串，然后查找是否为关系运算符；

(4) 对于注释的识别，当读入的 instring 出现"/"符号，继续读入，如果为"/"符号则判别为单行注释并直接跳过该行注释后面的代码，如果为"*"符号则判别为多行注释，并一直读取，直到出现"*/"为止；

(5) 对于浮点数的识别，除了能够正确读入浮点数外，还需判别 instring 中是否还会再出现小数点，若有多于一个小数点则判别为"Error"；

(6) 对于不正确的标识符的识别，当读入的 instring 中，字母后接数字则判别为标识符，若数字后面接字母或再接数字则判别为"Error"，即当读入数字时，若继续读入的是字母，或字母后面再接数字；

(7) 对于多个符号拼接的组合运算符的识别，程序通过划分各类特殊的情况，具体问题具体分析，予以识别与判定。

(8) 对于常数和标识符，为了能够顺利地进行语法分析，需要将识别到的常数和标识符进行保存，在此采用队列的数据结构。

(9) 对于不满足上述情况的文本，将判别为"Error"。

(10) 对于以上识别成功的文本，将经过一定的处理，如把 for 关键字改为 f，将常数或标识符改为 i，去除换行、空格、制表符等，并将所有字符拼接为字符串，用于接下来的语法分析。

(三) 语法分析器的实现

1. 文法设计

本次课程设计需要实现 FOR 语句的语法制导翻译器，需要为其设计相应的文法，使得设计的文法能够识别 FOR 语句的初始值、循环条件、循环增量、语句块等内容，具体的文法设计如下：

(1) S→f(NW)NX
(2) W→A;NB;ND
(3) A→i=E
(4) E→i
(5) B→i<i
(6) D→i++
(7) D→i--
(8) X→A;
(9) X→{M}

(10)M→MNA;
(11)M→A;
(12)N→ε
(13)E→E+E

其中，产生式(1)用于产生 FOR 语句的基本框架，FOR 语句的初始值、循环条件、循环增量由 W 产生，语句或语句块由 X 产生，即产生式(2)和(8)、(9)；对于产生式(2)，初始值由 A 产生，循环条件由 B 产生，循环增量由 X 产生，分别对应产生式(3)、(5)、(6)、(7)，此处常量与变量，包括字母和数字，统一用终结符 i 表示；在翻译过程时将会由词法分析器保存的名字代替，关键字 for 在文法分析时用终结符 f 代替，这是在词法分析后实现的，目的是为了规范终结符的长度为 1；对于语句或语句块的产生，对应于产生式(3)、(8)、(9)、(10)、(11)和(13)，通过这些产生式能够识别单条语句或语句块，其中语句主要为赋值语句或运算表达式。由于该文法需要基于具体的翻译模式，在产生式中插入产生空字的非终结符 N，用于在翻译模式设计时产生相应的语义动作。

构造好相应的 FOR 语句文法后，将基于 LR(1)文法的设计，采用自下向上的语法分析，从输入串开始，逐步进行“归约”，直到文法的开始符号。我们需要对该文法构造项目集族和预测分析表，然后设计语法分析的总控程序，使其能够利用设计好的预测分析表进行语法分析，分析的过程主要包括移进、归约、接受、报错四种。至此语法分析器设计完成。

2. 实现算法描述

对于 LR(1)文法分析的实现，我们需要考虑的内容如下。

(1) 项目集 I 闭包 CLOSURE(I)的构造

1) I 的任何项目都属于 CLOSURE(I)。

2) 若项目[A→α·Bβ, a]属于 CLOSURE(I)，B→ξ 是一个产生式，那么，对于 FIRST(βa) 中的每个终结符 b，如果[B→·ξ, b]原来不在 CLOSURE(I)中，则把它加进去。

3) 重复执行步骤 2)，直至 CLOSURE(I)不再增大为止。

在此次闭包的求解中涉及到了含有空串的产生式，当遇到含有空串的产生式时，也将其加入到闭包中。通过闭包的求解可以得到各个状态的项目集。

(2) GO 函数的构造

令 I 是一个项目集，X 是一个文法符号，函数 GO(I，X)定义为：

GO(I，X)＝CLOSURE(J)

其中 J＝{任何形如[A→αX·β, a]的项目| [A→α·Xβ, a]∈I}。

至此，可以构造识别活前缀的 DFA，最终构造的 DFA 详见附件。

(3) 分析表的构造

令每个 I_k 的下标 k 为分析表的状态，令含有[S′→·S, #]的 I_k 的 k 为分析器的初态。最终构造的预测分析表详见附件。

1) 若项目[A→α·aβ, b]属于 I_k 且 GO(I_k, a)＝I_j， a 为终结符，则置 ACTION[k, a]为 "s_j"。

2) 若项目[A→α·，a]属于 I_k，则置 ACTION[k, a]为“r_j”；其中假定 A→α为文法 G′的第 j 个产生式。

3) 若项目[S′→S·, #]属于 I_k，则置 ACTION[k, #]为 “acc”。

4) 若 GO(I_k，A)＝I_j，则置 GOTO[k, A]=j。

5) 分析表中凡不能用规则 1 至 4 填入信息的空白栏均填上“出错标志”。

值得注意的是，此次分析表构造的过程中会出现含有空串的项目，对于此类项目，需要执行相应的归约动作，故需要按照步骤 2)的方式执行。

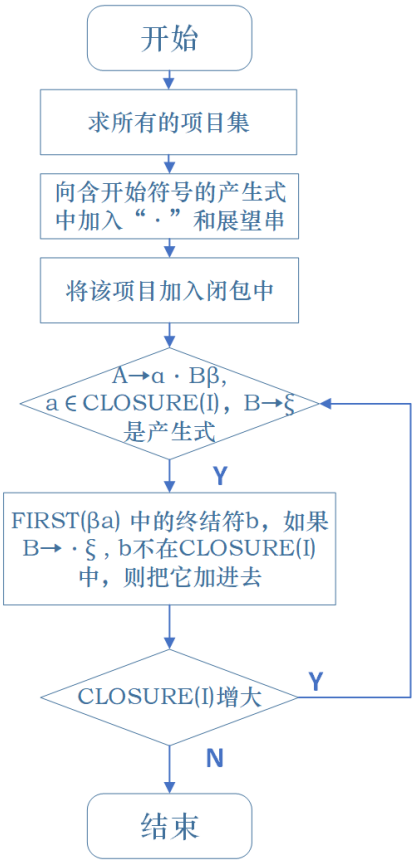


图 4 项目集 I 闭包 CLOSURE(I)的构造流程

(4) 语法分析过程

LR(1)的分析过程主要分为移进、归约、接受、报错四个动作：

1) 移进：当 table[i，a]= s_j 时，状态 j 移入到状态栈，把 a 移入到文法符号栈，其中 i, j 表示状态号。

2) 归约：当 table[i，a]＝r_k 时，如果栈顶形成句柄，则归约为相应的非终结符 A，即文法中有 A-B 的产生式，若 B 的长度为 R(即|B|=R)，则从状态栈和文法符号栈中自顶向下去掉 R 个符号，即栈指针 SP 减去 R，并把 A 移入文法符号栈内，j＝GOTO[i, A]移进状态栈，其中 i 为修改指针后的栈顶状态。

3) 接受：当归约到文法符号栈中只剩文法的开始符号 S 时，并且输入符号串已结束即当前输入符是'#'，则为分析成功，当 table 中该项为 acc 时分析成功接受。

4) 报错：当遇到状态栈顶为某一状态下出现不该遇到的文法符号时，则报错，说明输入端不是该文法能接受的符号串。

(四)语法制导翻译器的实现

1. 属性文法与翻译模式设计

经过文法及属性文法的设计，得到的分析器能够实现基本的语法分析，为了能够实现一遍扫描的翻译的功能，翻译器需要在语法分析的过程中执行相应的语义动作，对此在 FOR 语句文法的基础上，设计的翻译模式如下：

$S \rightarrow f(N_0W)N_1X$	{ backpatch(W.truelist,N1.quad); backpatch(X.nextlist,N3.quad); S.nextlist:=W.falselist emit('j, -, -,' N3.quad) }
$W \rightarrow A;N_2B;N_3D$	{ W.truelist:=B.truelist; W.falselist:=B.falselist; backpatch(D.nextlist,N2.quad); }
A→id=E	{ emit('=' ',' E.place ',' '-' ',' id.name) }
E→id	{ E.place:=id.name }
$B \rightarrow id_1$ relop id_2	{ B.truelist:=makelist(nextquad); B.faslelist:=makelist(nextquad+1); emit('j'relop.op', 'id1.name','id2.name',' '-'); emit('j, -, -, -') }
D→id++	{ D.nextlist:=makelist(nextquad+1); emit('+, id.name, 1, id.name') ; emit('j, -, -, -') }

D→id--	{ D.nextlist:=makelist(nextquad+1); emit('-, id.name, 1, id.name') ; emit('j, -, -, -') }
X→A;	{ A.nextlist:=makelist(); X.nextlist:=A.nextlist; }
X→{M}	{ X.nextlist:=M.nextlist }
M→M_1 N_4 A;	{ A.nextlist:=makelist(); M.nextlist:=A.nextlist; }
M→A;	{ A.nextlist:=makelist(); M.nextlist:=A.nextlist; }
N→ε	{ N.quad:=nextquad }
E→E_1+E_2	{ E.place:=newtemp; emit('+' ',' E1.place ',' E2.place ',' E.place) }

为了区分符号的属性，当产生式中出现重复的符号时，将加上标记。此处引入非终结符 N 用于执行语义动作，即记录下一条语句的位置，以便在回填时给出对应的语句位置，其中记录的位置有初始赋值、判断循环条件、循环增量、每条语句或语句块。

2. 实现算法描述

翻译过程中将根据终结符或非终结符所具有的属性，将其划分为 Undef、Sen、Ctl、Sys 四类，其具体属性如表 1 所示。

表 1 符号基本属性定义

类型	属性	描述
Undef	quad	记录下一条语句的位置
Sen	nextlist	记录需要回填下一条语句的语句
Ctrl	truelist; falselist	记录需要回填条件控制后跳转位置的语句
Sys	place	记录变量或常量的名字

为了能够正确进行语法制导翻译过程，在翻译的具体实现过程中，通过建立以上四类的栈，在归约的同时根据归约对应的产生式执行相应的语义动作，当语义动作中相应的符号出现在赋值号的右边时，则在执行语义动作前将对应的符号出栈，若出现在赋值号的左边时，则在执行语义动作之后将对应的符号压栈。

在执行语义动作的过程中可能需要输入在词法分析过程中得到的常量或变量，之前在词法分析的过程中是将这些符号保存在队列的数据结构中，由于归约是按照自左向右自底向上的方向进行，故当需要用到某符号的名字时只需要将队列元素出队即可。但值得注意的是，赋值语句的动作执行较为特殊，其需要先执

行赋值后右边的归约动作，而后再进行赋值的归约，在此若按照先前的思路直接将队列元素出队，则会造成顺序上的错误，故在执行赋值语句右边的归约动作时，将做以下调整：

```
E2 = SysStack.Pop();
E1 = SysStack.Pop();
string tmp = E1.place;
E1.place = E2.place;
E2.place = insys.Dequeue();
int cnt = insys.Count;
insys.Enqueue(tmp);
while (cnt-- > 0)
{
    string str = insys.Dequeue();
    insys.Enqueue(str);
}
```

按照归约的顺序，E_1 中保存的.place 属性实际上是赋值号左边的变量名，而 E_1 中保存的.place 属性实际上是赋值号右边的第一个符号名，故需要暂时将 E_1 中保存的.place 属性进行保存，并调整正确的值，由于将赋值号右边第二个符号出队后需要将赋值号左边的变量重新入队，并调整顺序，使该入队元素出现在队头。

此外，在制定翻译模式时还涉及到了几个函数，其具体作用如所示。

表 2 翻译模式中涉及到的函数解释

函数名称	传入参数	返回值	描述
emit	Quartrtte，四元式	无	输出四元式
makelist	int，下一条语句的位置	List<int>，产生的链表	创建回填链表
newtemp	无	string，产生的新变量名	产生新变量
backpatch	List<int>需要回填的链表； int 回填的位置	无	回填

以上函数的具体实现如下：

(1) emit 函数的实现

emit 函数主要用于输出四元式并使语句的位置加一，在此，四元式是通过类进行封装，在调用 emit 函数前，先对四元式中的数据项进行赋值，然后将其作为参数调用 emit 函数。

```
public class Quartrtte
{
    public int quad;          //当前四元式位置
```

```
        public string op;       //运算符
        public string arg1;     //操作数1
        public string arg2;     //操作数2
        public string res;      //结果
    }
```

在此所设计的 emit 函数并非直接将其输出，而是保存在四元式列表 List<Quartrtte> QuarList 中，最后统一由相关函数进行打印输出。

(2) makelist 函数的实现、

makelist 函数需要创建相应的链表并返回，在此只需利用 C#中 List 变量，将函数输入的 quad 加入到链表中并返回，此外，list 的设置实际上用于保存需要回填的位置，因此会涉及到链表的合并，然而在此次设计的翻译模式中链表只涉及到了一个位置，在实际实现并未考虑链表合并的操作。

(3) newtemp 函数的实现

newtemp 的函数主要返回一个新变量，要求返回的变量与先前的变量不重复，实现并不困难，只需要内部设置一个用于记录个数的变量，每调用一次该变量的值加 1，与 emit 函数的用法类似，最后返回 string 类型的变量名即可。

(4) backpatch 函数的实现

该函数目的是找到链表中对应的四元式位置，并将目标位置填到四元式中。由于在此是将四元式统一保存在 List<Quartrtte> QuarList 中，因此只需遍历该 QuarList，找到对应的四元式，并修改其.res 属性即可。

```
    void backpatch(List<int> list, int quad)
    {
        foreach (var q in QuarList)
        {
            foreach (var l in list)
            {
                if (q.quad == l)
                {
                    q.res = quad.ToString();
                }
            }
        }
    }
```

至此，语法制导翻译器实现完毕。该翻译器能够识别各类符号，并进行语法分析，在分析的同时执行相应的语义动作，最后生成四元式。

四、 设计的输入和输出形式

本次课程设计采用图形化用户界面的方式呈现，如图 5 所示，用户只需将需要分析的源程序保存在文本文件中，然后在分析程序中找到该文件并打开。然后点击判断即可。

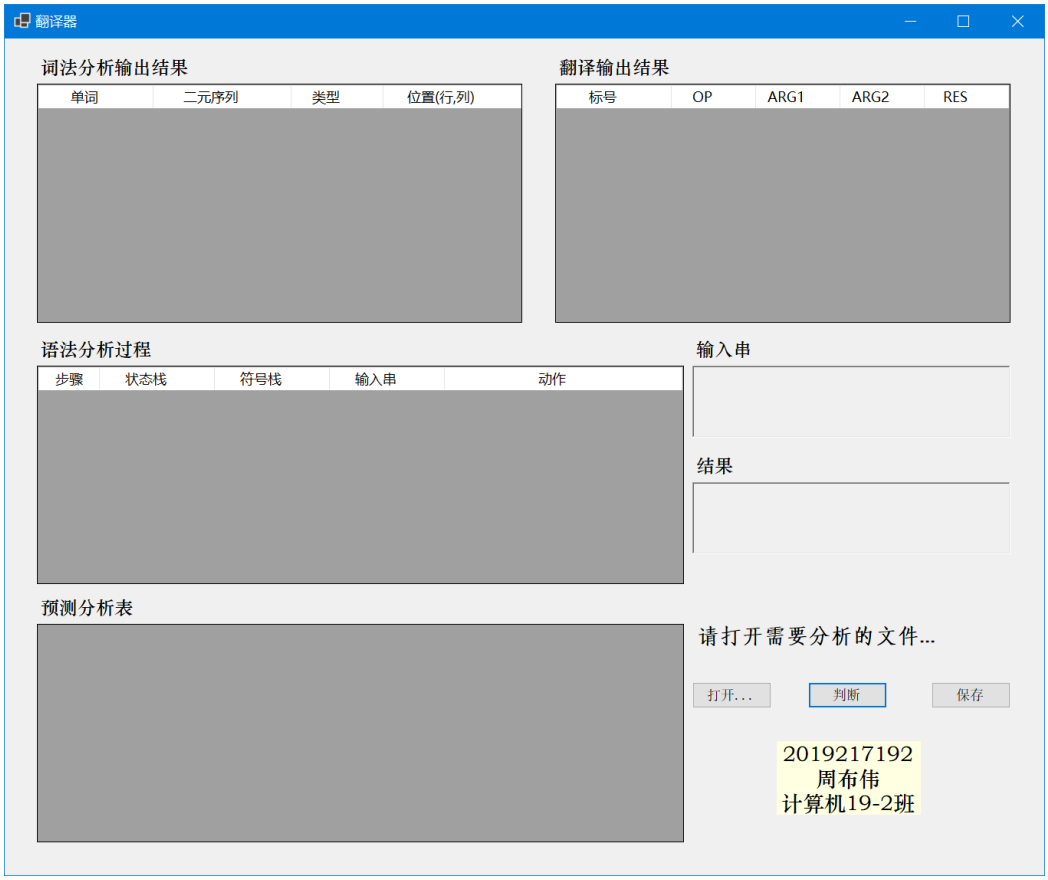


图 5 程序初始化界面

分析过程中产生的中间结果将通过程序中的控件显示，如图 6 所示，分别有词法分析输出结果、输入串、语法分析过程、预测分析表、翻译输出结果、最终结果等。其中：

词法分析输出结果以“单词-二元序列-类型-位置”的形式输出词法分析器在分析过程中所识别到的单词。

输入串是将词法分析过程中遇到的符号经过去除空格、换行、制表符，替换关键字，简化标识符与常数等措施所得到的结果。

语法分析过程主要包括步骤、状态栈、符号栈、输出串、动作的输出，当词法分析或语法分析出现错误时会在分析结果中标红显示，并停止后续的分析过程，

如图 7 所示。

图 6 分析结果显示

语法分析的同时会显示预测分析表，预测分析表由状态作为行，终结符、非终结符作为列构成，分别形成 ACTION 表和 GOTO 表。值得一提的是，由于本次设计过程中文法指定，因此在实际实现时可以避免反复求解识别活前缀的 DFA，对此，为了提高程序的运行速度，结合《编译原理》课程实验 3——LR 语法分析设计的成果，在此次设计前，先将设计的文法通过该程序生成项目集族和预测分析表，然后将得到的结果内置到本次设计程序中，因此在分析过程中就无需反复求解项目集族和预测分析表。

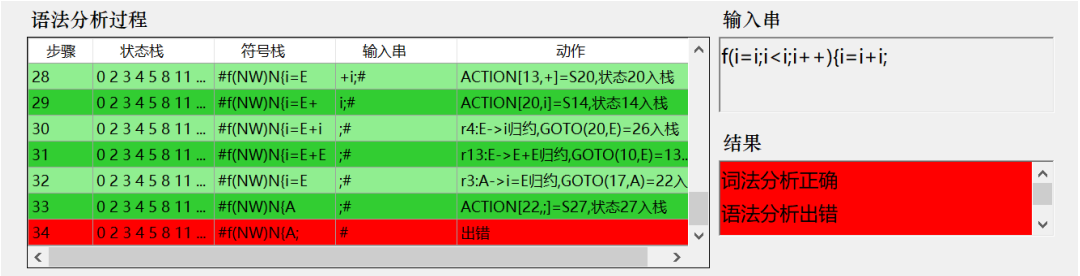


图 7 分析过程中发现错误

翻译输出结果由标号、OP(操作符)、ARG1(操作数 1)、ARG2(操作数 2)、RES(结果)构成四元式，能够识别输出赋值语句、条件控制语句等的四元式。

点击“保存”，上述分析的结果将以文件的形式输出，其格式内容与上述分析结果相同。当需要多次输出时能够产生多个文件，并保证命名不重复。

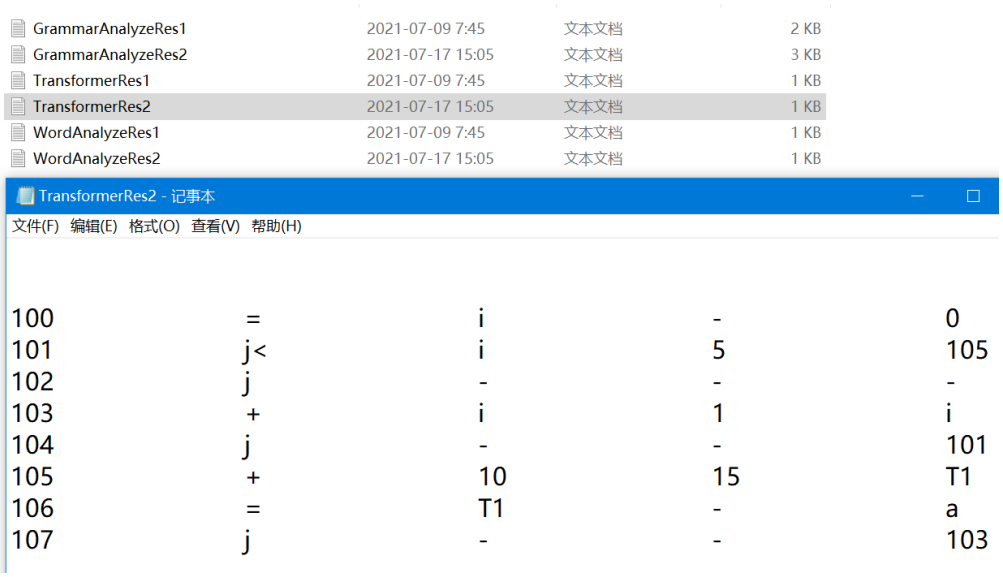


图 8 分析结果输出

五、 程序运行的结果

首先，测试如图 9 所示的源程序代码。由图可见代码中含有注释、由语句块包含的多行语句。

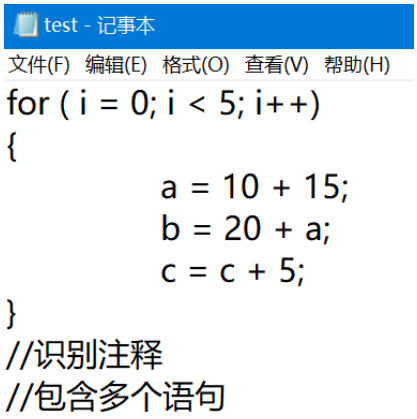


图 9 测试样例 1

从词法分析结果中可以看出，本程序能够正确识别关键字、标识符、分界符、常量等，并输出其中的二元序列、位置等信息；从词法分析后的输入串可以看出，已经将遇到的输入源程序去除空格、换行、制表符，替换关键字，简化标识符、运算符与常数；语法分析结果中可以看出，语法分析的总控程序能够正确地完成移进、归约、接受、报错(如上文所示)的动作；翻译结果可以看出，翻译程序能

够正确执行相应的语义动作，并输出相应的四元式。

词法分析输出结果

单词	二元序列	类型	位置(行,列)
for	(1,for)	关键字	(1, 1)
(	(2,()	分界符	(1, 2)
i	(6,i)	标识符	(1, 3)
=	(3,=)	算术运算符	(1, 4)
0	(5,0)	常数	(1, 5)
;	(2,;)	分界符	(1, 6)
i	(6,i)	标识符	(1, 7)
<	(4,<)	关系运算符	(1, 8)

图 10 样例 1 词法分析结果

输入串

f(i=i;i<i;i++){i=i+i;i=i+i;i=i+i;}

图 11 词法分析后产生的输入串

语法分析过程

步骤	状态栈	符号栈	输入串	动作
56	0 2 3 4 5 8 11 ...	#f(NW)N{MNi...	;}#	r3:A->i=E归约,GOTO(28,A)=32入
57	0 2 3 4 5 8 11 ...	#f(NW)N{MNA	;}#	ACTION[32,;]=S35,状态35入栈
58	0 2 3 4 5 8 11 ...	#f(NW)N{MNA;	}#	r10:M->MNA;归约,GOTO(17,M)=
59	0 2 3 4 5 8 11 ...	#f(NW)N{M	}#	ACTION[23,}]=S29,状态29入栈
60	0 2 3 4 5 8 11 ...	#f(NW)N{M}	#	r9:X->{M}归约,GOTO(11,X)=16入
61	0 2 3 4 5 8 11 ...	#f(NW)NX	#	r1:S->f(NW)NX归约,GOTO(0,S)=1
62	0 1	#S	#	Acc：分析成功

图 12 样例 1 语法分析结果

翻译输出结果

标号	OP	ARG1	ARG2	RES
100	=	i	-	0
101	j<	i	5	105
102	j	-	-	-
103	+	i	1	i
104	j	-	-	101
105	+	10	15	T1
106	=	T1	-	a
107	+	20	a	T2

图 13 样例 1 翻译结果

然后，更换样例，改变测试初始值、循环条件、循环增量，并将语句块改为单个语句。最后结果如图 15 所示。观察输入串，发现程序中的“>”被更改为“<”，这同样是为了便于语法分析的方便与统一而进行的操作，翻译结果依然正确。

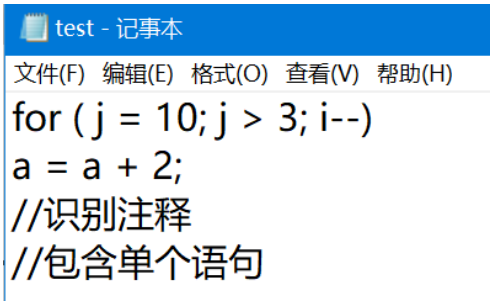


图 14 测试样例 2

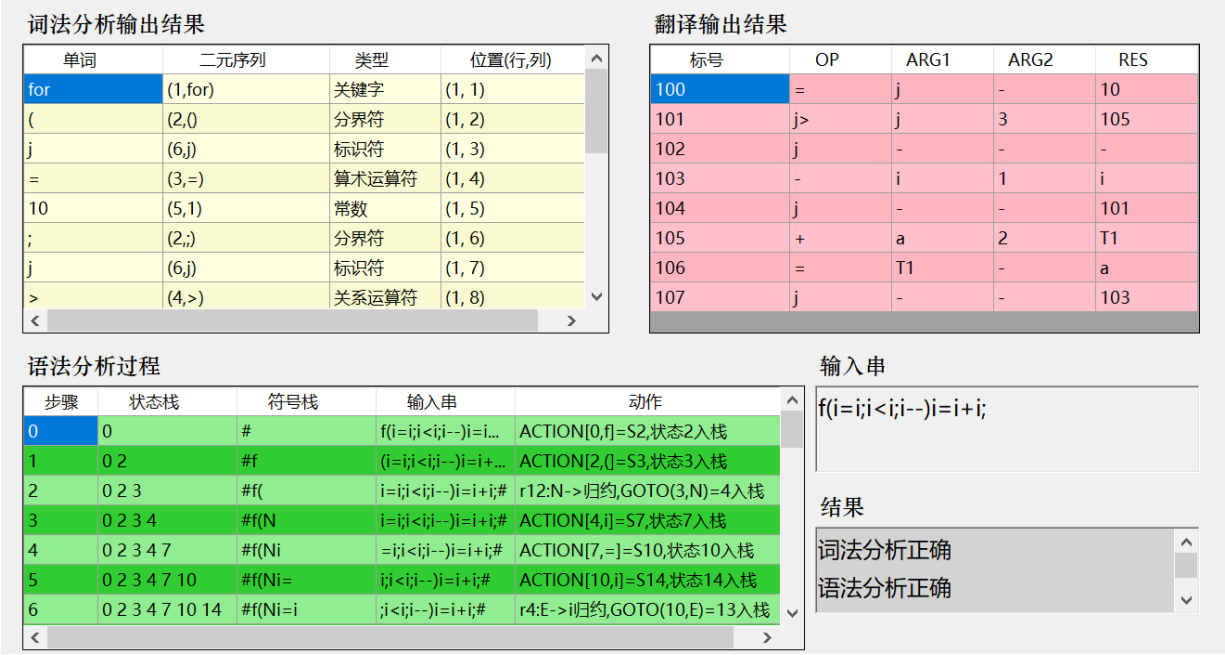


图 15 样例 2 输出结果

最后，选择错误样例，去掉样例 2 中结尾的分号，从结果中可以看出本程序能正确识别错误，并给出错误提示，同时停止后续的分析过程。

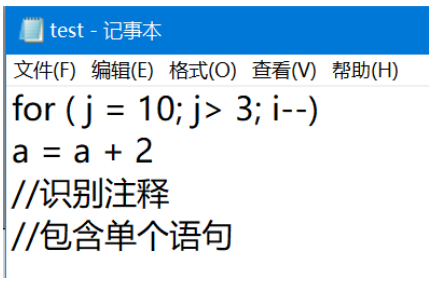


图 16 测试样例 3

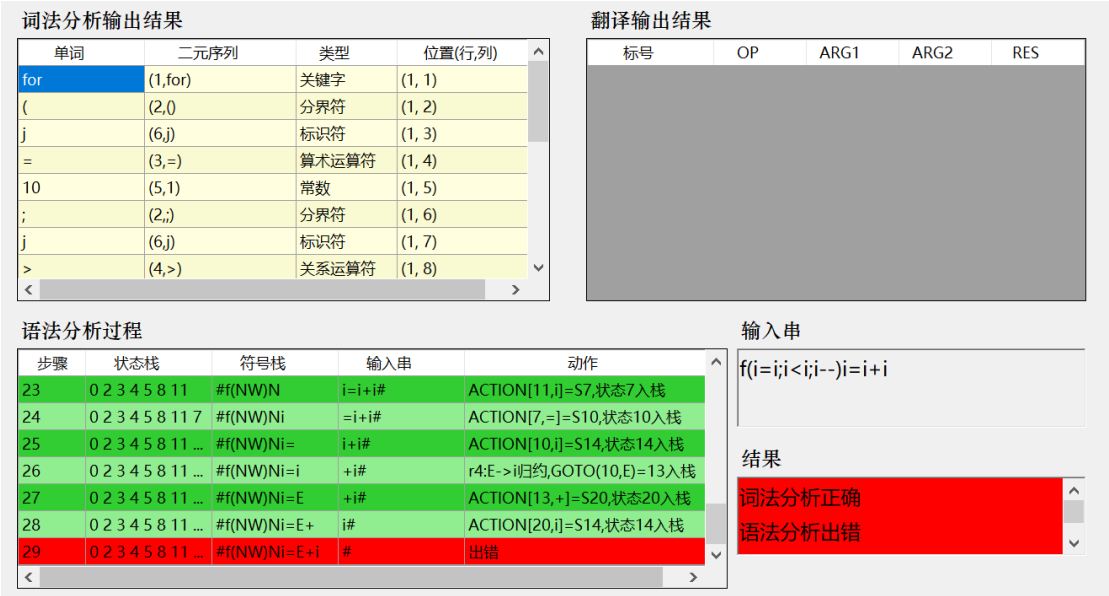


图 17 测试样例 3 输出结果

六、 总结与体会

本次课程设计实现了 FOR 语句的语法制导翻译器，其中包含了词法分析器、语法分析器、语义分析和中间代码生成器，本程序依托 C++语言的语法与命名习惯，识别输入文本中的 C++语言代码，判断输入符号类型，产生二元序列和位置坐标；采用 LR(1)文法进行 FOR 循环语句的设计，同时考虑翻译模式。由最终语法分析器的总控程序通过得到的预测分析表对输入串进行自底向上分析。翻译模式包括赋值语句、条件控制语句的生成等，为每条产生式配备语义子程序，在翻译模式的基础上，结合语法分析器，当语法分析过程出现归约时执行相应的语义动作。最终翻译的结果以四元式的形式给出。

本次采用 C#语言进行设计，并实现了图形化用户界面的交互方式，能够进行文件的输入输出，并展现结果。虽然在课内并未介绍 C#语言，对其内部对象、属性、方法还很陌生，但是并不妨碍程序的设计，利用 Visual Studio 2019 能够很好地通过图形化交互的方式对程序窗口进行设计，其内部的部分语句或数据类型，如 foreach、List 等，相较 C++有了一定的优化，使用起来十分方便。

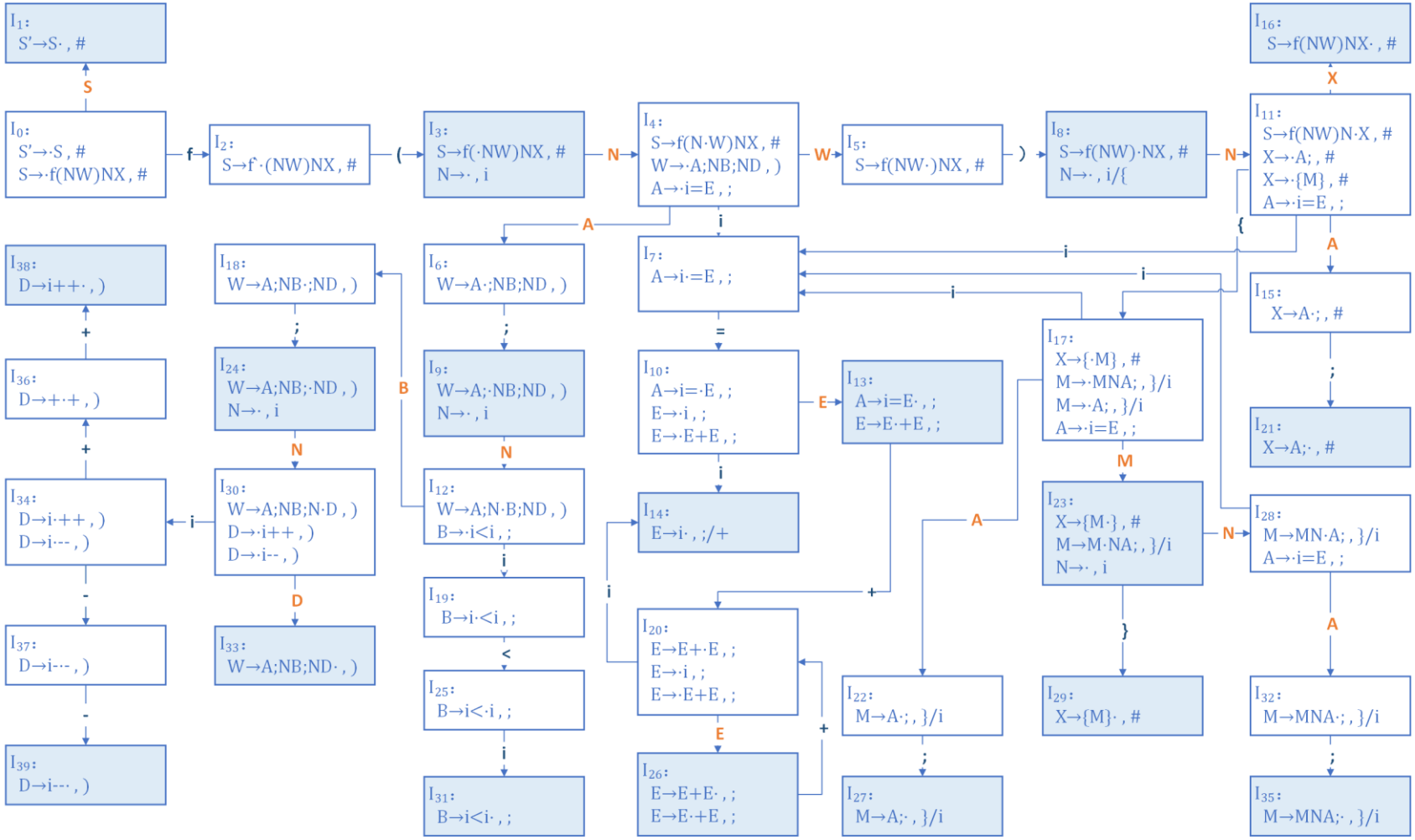
此次课程设计是基于《编译原理》课程的学习以及课程实验实现的，先前的课程实验只是对各类分析器分别设计，并未将其进行组合，在此次课程设计中就需要对其进行组合，在组合过程中就需要考虑很多因素，如词法分析过程中得到的结果需要经过去除空格、换行、制表符，更改关键词、常数、变量名等操作后

才能作为输入串进行语法分析，同时词法分析时需要保存常数、变量名等信息，以便翻译过程中的使用。

本次课程设计中涉及到的文法和翻译模式，全部自主设计，为了防止用程序生成 DFA 和预测分析表出错，又通过手工演算的方式进行验证，演算内容详见附件。在自主设计的过程中，我对编译的基本原理有了更加深刻的认识，效果比单纯从课堂中被动地吸收知识要好很多，在自主设计的过程中会发现很多的问题，同时也对求识别活前缀的 DFA、预测分析表等算法有了更加深刻的理解与感悟，而不是停留于死记硬背的层面。

此次课程设计比以往的任何实验或课程设计要复杂和困难，不仅是知识理解上的困难也是代码实现的困难，一开始总会有无从下手的感觉，但经过课堂的学习、实验的积累，到最后也能很顺利地实现需要的功能。在课程设计的过程中，代码的编写和调试的经验和能力也在不断提升。在未来的学习过程中，仍需要进一步提升这几个方面的能力。

附件 1：识别活前缀的 DFA



附件 2：LR(1)预测分析表

	for	(	)	i	=	<	;	+	-	{	}	#	S	W	M	X	E	N	A	D	B
0	s2												1								
1												acc									
2		s3																			
3				r12														4			
4				s7										5					6		
5			s8																		
6							s9														
7					s10																
8				r12						r12								11			
9				r12														12			
10				s14													13				
11				s7						s17						16			15		
12				s19																	18
13							r3	s20													
14							r4	r4													
15							s21														
16												r1									
17				s7											23				22		
18							s24														
19						s25															
20				s14													26				
21												r8									
22							s27														
23				r12							s29							28			
24				r12														30			
25				s31																	
26							r13	s20													
27				r11							r11										
28				s7															32		
29												r9									
30				s34																33	
31							r5														
32							s35														
33			r2																		
34								s36	s37												
35				r10							r10										
36								s38													
37									s39												
38			r6																		
39			r7																		

附件 3：源程序清单（部分核心代码）

1. 词法分析器

```
public void Analyze(StreamReader sr)
        {
            //System.Console.Clear();
            System.Console.WriteLine("单词\t\t二元序列\t\t类型\t\t位置(行，列)");
            string instring;
            char ch;
            int loc = 0;
            outstring = "";
            logList.Clear();

            while (!sr.EndOfStream)
            {
                Log l = new Log();
                l.strString = "";
                ch = (char)sr.Read();
                loc++;
                instring = "";

                //判断空格换行
                if (ch == ' ' || ch == '\t' || ch == '\n')
                {
                    if (ch == '\n')
                    {
                        row++;
                        col = 0;
                    }
                }
                //判断注释
                else if (ch == '/')
                {
                    instring = ch.ToString();
                    ch = (char)sr.Read();
                    loc++;
                    if (ch == '/')
                    {
                        while (!sr.EndOfStream)
                        {
                            ch = (char)sr.Read();
                            loc++;
                            if (ch == '\n')
                            {
                                row++;
                                col = 0;
                                break;
                            }
                        }
                    }
                    else if (ch == '*')
                    {
                        while (!sr.EndOfStream)
                        {
```

```
                                ch = (char)sr.Read();
                                loc++;
                                if (ch == '*')
                                {
                                    ch = (char)sr.Read();
                                    loc++;
                                    if (ch == '/')
                                        break;
                                    else if (ch == '\n')
                                    {
                                        row++;
                                        col = 0;
                                    }
                                }
                                else if (ch == '\n')
                                {
                                    row++;
                                    col = 0;
                                }
                            }
                        }
                        else
                        {
                            col++;
                            l.strString = instring;
                            l.strPair = "(3," + instring + ")";
                            l.strType = "算术运算符";
                            l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                            System.Console.WriteLine(instring + "\t\t" + "(3," + instring + ")\t\t\t" + "算术运算符\t(" +
row.ToString() + "," + col.ToString() + ")");
                            sr.DiscardBufferedData();
                            loc -= 1;
                            sr.BaseStream.Seek(loc, SeekOrigin.Begin);
                        }
                    }
                    //判断是否为关键字标识符
                    else if (IsLetter(ch))
                    {
                        while (IsLetter(ch) || IsDigit(ch))
                        {
                            instring = instring + ch;
                            ch = (char)sr.Read();
                            loc++;
                        }
                        if (IsKey(instring))//识别关键字
                        {
                            col++;
                            l.strString = instring;
                            l.strPair = "(1," + instring + ")";
                            l.strType = "关键字";
                            l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                            System.Console.WriteLine(instring + "\t\t" + "(1," + instring + ")\t\t\t" + "关键字\t\t(" + row.ToString()
+ "," + col.ToString() + ")");
                            if (instring == "for")    outstring += "f";
                        }
                        else//识别标识符
                        {
                            col++;
```

```
                        if (!IsInId(instring))
                        {
                            int ptr;
                            ptr = InsertId(instring);
                        }
                        l.strString = instring;
                        l.strPair = "(6," + instring + ")";
                        l.strType = "标识符";
                        l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                        System.Console.WriteLine(instring + "\t\t" + "(6," + instring + ")\t\t\t" + "标识符\t\t(" + row.ToString()
+ "," + col.ToString() + ")");
                        sys.Enqueue(instring);
                        outstring += "i";
                    }
                    sr.DiscardBufferedData();
                    loc -= 1;
                    sr.BaseStream.Seek(loc, SeekOrigin.Begin);

                }
                //判断是否为常数
                else if (IsDigit(ch))
                {
                    while (IsDigit(ch) || ch == '.')
                    {
                        instring = instring + ch;
                        ch = (char)sr.Read();
                        loc++;
                    }
                    if (IsLetter(ch))//判断数字后是否紧跟字符，是则报错
                    {
                        while (IsLetter(ch) || IsDigit(ch))
                        {
                            instring = instring + ch;
                            ch = (char)sr.Read();
                            loc++;
                        }
                        col++;
                        l.strString = instring;
                        l.strPair = "Error";
                        l.strType = "Error";
                        l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                        System.Console.WriteLine(instring + "\t\t" + "Error\t\t\t" + "Error\t\t(" + row.ToString() + "," +
col.ToString() + ")");
                    }
                    else
                    {
                        int num = fingPoint(instring);
                        if (num == 0)
                        {
                            string bin = "";
                            bin = Dec2Bin(instring);
                            if (!IsInConst(bin))
                            {
                                int ptr;
                                ptr = InsertConst(bin);
                            }
                            col++;
                            l.strString = instring;
```

```
                                l.strPair = "(5," + bin + ")";
                                l.strType = "常数";
                                l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                                System.Console.WriteLine(instring + "\t\t" + "(5," + bin + ")\t\t\t" + "常数\t\t(" + row.ToString() +
"," + col.ToString() + ")");
                                sys.Enqueue(instring);
                                outstring += "i";
                            }
                            else if (num == 1)
                            {
                                col++;
                                l.strString = instring;
                                l.strPair = "(5," + instring + ")";
                                l.strType = "浮点数";
                                l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                                System.Console.WriteLine(instring + "\t\t" + "(5," + instring + ")\t\t\t" + "浮点数\t\t(" +
row.ToString() + "," + col.ToString() + ")");
                                sys.Enqueue(instring);
                                outstring += "i";
                            }
                            else
                            {
                                col++;
                                l.strString = instring;
                                l.strPair = "Error";
                                l.strType = "Error";
                                l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                                System.Console.WriteLine(instring + "\t\t" + "Error\t\t" + "Error\t\t(" + row.ToString() + "," +
col.ToString() + ")");
                            }
                        }
                        sr.DiscardBufferedData();
                        loc -= 1;
                        sr.BaseStream.Seek(loc, SeekOrigin.Begin);
                    }
                    //判断是否为符号
                    else if (s2.Contains(ch.ToString()) || s3.Contains(ch.ToString()) || s1.Contains(ch.ToString()))
                    {
                        bool ins3 = false;
                        while (s3.Contains(ch.ToString()))
                        {
                            instring += ch.ToString();
                            ch = (char)sr.Read();
                            loc++;
                            ins3 = true;
                        }
                        if (ins3)
                        {
                            sr.DiscardBufferedData();
                            loc -= 1;
                            sr.BaseStream.Seek(loc, SeekOrigin.Begin);
                        }
                        //是否为分界符和算数运算符
                        else
                        {
                            instring = ch.ToString();
                            //System.Console.WriteLine();
                            //ch = (char)sr.Read();
```

```
                        }
                        //判断分界符
                        if (s1.Contains(instring))
                        {
                            col++;
                            l.strString = instring;
                            l.strPair = "(2," + instring + ")";
                            l.strType = "分界符";
                            l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                            System.Console.WriteLine(instring + "\t\t" + "(2," + instring + ")\t\t\t" + "分界符\t\t(" + row.ToString()
+ "," + col.ToString() + ")");
                            outstring += instring;
                        }
                        //判断算术运算符
                        else if (s2.Contains(instring))
                        {
                            col++;
                            l.strString = instring;
                            l.strPair = "(3," + instring + ")";
                            l.strType = "算术运算符";
                            l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                            System.Console.WriteLine(instring + "\t\t" + "(3," + instring + ")\t\t\t" + "算术运算符\t(" +
row.ToString() + "," + col.ToString() + ")");
                            outstring += instring;

                        }
                        //判断关系运算符
                        else if (s3.Contains(instring))
                        {
                            col++;
                            l.strString = instring;
                            l.strPair = "(4," + instring + ")";
                            l.strType = "关系运算符";
                            l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                            System.Console.WriteLine(instring + "\t\t" + "(4," + instring + ")\t\t\t" + "关系运算符\t(" +
row.ToString() + "," + col.ToString() + ")");
                            sys.Enqueue(instring);
                            outstring += "<";
                        }
                        else
                        {
                            col++;
                            l.strString = instring;
                            l.strPair = "Error";
                            l.strType = "Error";
                            l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                            System.Console.WriteLine(instring + "\t\t" + "Error\t\t" + "Error\t\t(" + row.ToString() + "," +
col.ToString() + ")");

                        }
                    }
                    else if (instring != "")
                    {
                        col++;
                        instring = ch.ToString();
                        l.strString = instring;
                        l.strPair = "Error";
                        l.strType = "Error";
```

```
                    l.strLocation = "(" + row.ToString() + ", " + col.ToString() + ")";
                    System.Console.WriteLine(instring+"\t\t" + "Error\t\t\t" + "Error\t\t(" + row.ToString() + "," +
col.ToString() + ")");
                }
                if (l.strString != "")    logList.Add(l);
            }
        }
```

2. 语法分析器

```
public void LR1()
        {
            initial();
            int s;//状态栈栈顶
            char a; //ip所指向的符号
            int sp = 0;//几号状态
            string right = "";//产生式右部
            Transformer ts = new Transformer();
            ts.setSys(outsys);
            while (true)
            {
                s = statusStack.First();
                //System.Console.WriteLine(inString + ip.ToString());
                a = inString[ip];
                int index = getIndex(a);

                //tbRes.Text = index.ToString();
                System.Console.WriteLine(s.ToString()+"    "+index.ToString());
                if (index < 0 || ACTION[s][index] == null)
                {
                    //a不是终结符或者action表中没有对应信息
                    Error();
                    break;
                }
                else
                {
                    if (ACTION[s][index] == "acc")
                    {
                        //成功接收
                        tbRes.Text = tbRes.Text.Insert(tbRes.Text.Length, "语法分析正确\n");
                        WriteLog(1);
                        break;
                    }
                    else if (ACTION[s][index][0] == 's')
                    {
                        //移进动作
                        int next = Convert.ToInt32(ACTION[s][index].Remove(0, 1));//查找动作表，移进后转到第几个状态
                        WriteLog(2, next, s, a);

                        symbolStack.Push(a);
                        statusStack.Push(next);
                        ++ip;

                    }
                    else if (ACTION[s][index][0] == 'r')
                    {
                        //归约动作
                        int n = Convert.ToInt32(ACTION[s][index].Remove(0, 1));//查找动作表，用第几个产生式归约
                        right = GetRight(n);//获取第n个产生式的右部
```

```
                    sp = statusStack.ElementAt(right.Length);//几号状态
                    int k = GOTO[sp][getNIndex(production[n][0])];//查找goto表，转到第几号状态
                    WriteLog(3, n, sp, production[n][0], k);

                    for (int i = 0; i < right.Length; i++)
                    {
                        symbolStack.Pop();
                        statusStack.Pop();
                    }

                    statusStack.Push(k);
                    symbolStack.Push(production[n][0]);

                    ts.Action(n);
                }
            }
        }
        if (isGoOn)
        {
            ShowTransProcedure(ts);
        }

    }
```

3. 翻译器

```
    void emit(Quartrtte qua)
    {
        QuarList.Add(qua);
        nextquad++;
    }

    List<int> makelist(int quad)
    {
        List<int> l = new List<int>();

        l.Add(quad);

        return l;
    }
    string newtemp()
    {
        tempNum++;
        return "T" + tempNum.ToString();
    }


    void backpatch(List<int> list, int quad)
    {
        foreach (var q in QuarList)
        {
            foreach (var l in list)
            {
                if (q.quad == l)
                {
                    q.res = quad.ToString();
                }
            }
        }
```

```
}

List<int> merge(List<int> l1, List<int> l2)
{
    List<int> l = new List<int>();

    foreach (var item in l1)
    {
        foreach (var jtem in l2)
        {
            if (!l.Contains(item))
            {
                l.Add(item);
            }
            if (!l.Contains(jtem))
            {
                l.Add(jtem);
            }
        }
    }

    return l;
}

void Action1(Sen X, Ctrl W, Sen S, Undef N3, Undef N1)
{
    backpatch(X.nextlist, N3.quad);
    backpatch(W.truelist, N1.quad);
    S.nextlist = W.falselist;
    Quartrtte q = new Quartrtte();
    q.quad = nextquad;
    q.op = "j";
    q.arg1 = "-";
    q.arg2 = "-";
    q.res = N3.quad.ToString();
    emit(q);
}

void Action2(Ctrl W, Ctrl B, Sen D, Undef N)
{
    W.truelist = B.truelist;
    W.falselist = B.falselist;
    backpatch(D.nextlist, N.quad);
}

void Action3(Sys E)
{
    Quartrtte q = new Quartrtte();
    q.quad = nextquad;
    q.op = "=";
    q.arg1 = E.place;
    q.arg2 = "-";
    q.res = insys.Dequeue();
    emit(q);
}

void Action4(Sys E)
{
```

```
        E.place = insys.Dequeue();
}

void Action5(Ctrl B)
{
        B.truelist = makelist(nextquad);
        B.falselist = makelist(nextquad + 1);

        Quartrtte q1 = new Quartrtte();
        q1.quad = nextquad;
        q1.arg1 = insys.Dequeue();
        q1.op = "j" + insys.Dequeue();
        q1.arg2 = insys.Dequeue();
        q1.res = "-";
        emit(q1);
        Quartrtte q2 = new Quartrtte();
        q2.quad = nextquad;
        q2.op = "j";
        q2.arg1 = "-";
        q2.arg2 = "-";
        q2.res = "-";
        emit(q2);
}

void Action6(Sen D)
{
        D.nextlist = makelist(nextquad + 1);

        Quartrtte q1 = new Quartrtte();
        q1.quad = nextquad;
        q1.arg1 = insys.Dequeue();
        q1.op = "+";
        q1.arg2 = "1";
        q1.res = q1.arg1;
        emit(q1);
        Quartrtte q2 = new Quartrtte();
        q2.quad = nextquad;
        q2.op = "j";
        q2.arg1 = "-";
        q2.arg2 = "-";
        q2.res = "-";
        emit(q2);
}

void Action7(Sen D)
{
        D.nextlist = makelist(nextquad + 1);

        Quartrtte q1 = new Quartrtte();
        q1.quad = nextquad;
        q1.arg1 = insys.Dequeue();
        q1.op = "-";
        q1.arg2 = "1";
        q1.res = q1.arg1;
        emit(q1);
        Quartrtte q2 = new Quartrtte();
        q2.quad = nextquad;
        q2.op = "j";
```

```
        q2.arg1 = "-";
        q2.arg2 = "-";
        q2.res = "-";
        emit(q2);
    }

    void Action8(Sen X, Sen A)
    {
        A.nextlist = makelist(nextquad);
        X.nextlist = A.nextlist;

    }

    void Action9(Sen X, Sen M)
    {
        X.nextlist = M.nextlist;
    }

    void Action10(Sen M1, Sen M, Sen A, Undef N)
    {
        A.nextlist = makelist(nextquad);
        //backpatch(M1.nextlist, N.quad);
        M.nextlist = A.nextlist;
    }

    void Action11(Sen M, Sen A)
    {
        A.nextlist = makelist(nextquad);
        M.nextlist = A.nextlist;
    }

    void Action12(Undef N)
    {
        N.quad = nextquad;
    }

    void Action13(Sys E, Sys E1, Sys E2)
    {
        E.place = newtemp();
        Quartrtte q = new Quartrtte();
        q.quad = nextquad;
        q.op = "+";
        q.arg1 = E1.place;
        q.arg2 = E2.place;
        q.res = E.place;
        emit(q);
    }

    public void setSys(Queue<string> sys)
    {
        insys = sys;
        foreach (var item in insys)
        {
            System.Console.Write(item + "    ");
        }
        System.Console.WriteLine();
    }
```

```
public void Action(int n)
{
    Ctrl W = new Ctrl();
    Ctrl B = new Ctrl();
    Sen X = new Sen();
    Sen M = new Sen();
    Sen M1 = new Sen();
    Sen A = new Sen();
    Sen D = new Sen();
    Sen S = new Sen();
    Sys E = new Sys();
    Sys E1 = new Sys();
    Sys E2 = new Sys();
    Undef N = new Undef();
    Undef N0 = new Undef();
    Undef N1 = new Undef();
    switch (n)
    {
        case 1:
            N1 = NStack.Pop();
            N0 = NStack.Pop();
            X = SenStack.Pop();
            W = CtlStack.Pop();
            Action1(X, W, S, N0, N1);
            SenStack.Push(S);
            break;
        case 2:
            B = CtlStack.Pop();
            D = SenStack.Pop();
            N1 = NStack.Pop();
            N0 = NStack.Pop();
            Action2(W, B, D, N0);
            CtlStack.Push(W);
            NStack.Push(N1);
            break;
        case 3:
            E = SysStack.Pop();
            Action3(E);
            break;
        case 4:
            Action4(E);
            SysStack.Push(E);
            break;
        case 5:
            Action5(B);
            CtlStack.Push(B);
            break;
        case 6:
            Action6(D);
            SenStack.Push(D);
            break;
        case 7:
            Action7(D);
            SenStack.Push(D);
            break;
        case 8:
            Action8(X, A);
            SenStack.Push(X);
```

```
                    break;
                case 9:
                    M = SenStack.Pop();
                    Action9(X, M);
                    SenStack.Push(X);
                    break;
                case 10:
                    M1 = SenStack.Pop();
                    N = NStack.Pop();
                    Action10(M1, M, A, N);
                    SenStack.Push(M);
                    break;
                case 11:
                    Action11(M, A);
                    SenStack.Push(M);
                    break;
                case 12:
                    Action12(N);
                    NStack.Push(N);
                    break;
                case 13:
                    E2 = SysStack.Pop();
                    E1 = SysStack.Pop();
                    string tmp = E1.place;
                    E1.place = E2.place;
                    E2.place = insys.Dequeue();
                    int cnt = insys.Count;
                    insys.Enqueue(tmp);
                    while (cnt-- > 0)
                    {
                        string str = insys.Dequeue();
                        insys.Enqueue(str);
                    }
                    Action13(E, E1, E2);
                    SysStack.Push(E);
                    break;
            }
        }
```

4. 文件的输入输出

```
private void btnOpen_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();

            ofd.InitialDirectory = "C:\\Users\\hw\\Desktop\\编译原理课程设计";
            ofd.Filter = "txt files (*.txt)|*.txt";
            ofd.RestoreDirectory = true;

            if (ofd.ShowDialog() == DialogResult.OK)
            {
                strReadFilePath = ofd.FileName;
                lbPath.Text = strReadFilePath.Substring(strReadFilePath.LastIndexOf("\\") + 1);
            }
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            FileStream fs;
```

```
                StreamWriter sw;

                if (isGoOn)
                {
                    for (int i = 1; ;i++)
                    {
                        if (!System.IO.File.Exists("C:\\Users\\hw\\Desktop\\编译原理课程设计\\输出文件
\\WordAnalyzeRes"+i.ToString()+".txt"))
                        {
                            //没有则创建这个文件
                            fs = new FileStream("C:\\Users\\hw\\Desktop\\编译原理课程设计\\输出文件\\WordAnalyzeRes" +
i.ToString() + ".txt", FileMode.Create, FileAccess.Write);//创建写入文件
                            sw = new StreamWriter(fs);
                            sw.WriteLine();
                            sw.WriteLine();
                            foreach (var item in WordAnalyzeRes)
                            {
                                sw.WriteLine(item);
                            }
                            sw.Close();
                            fs.Close();
                            break;
                        }
                    }

                    for (int i = 1; ; i++)
                    {
                        if (!System.IO.File.Exists("C:\\Users\\hw\\Desktop\\编译原理课程设计\\输出文件\\GrammarAnalyzeRes" +
i.ToString() + ".txt"))
                        {
                            //没有则创建这个文件
                            fs = new FileStream("C:\\Users\\hw\\Desktop\\编译原理课程设计\\输出文件\\GrammarAnalyzeRes" +
i.ToString() + ".txt", FileMode.Create, FileAccess.Write);//创建写入文件
                            sw = new StreamWriter(fs);
                            sw.WriteLine();
                            sw.WriteLine();
                            foreach (var item in GrammarAnalyzeRes)
                            {
                                sw.WriteLine(item);
                            }
                            sw.Close();
                            fs.Close();
                            break;
                        }
                    }


                    for (int i = 1; ; i++)
                    {
                        if (!System.IO.File.Exists("C:\\Users\\hw\\Desktop\\编译原理课程设计\\输出文件\\TransformerRes" +
i.ToString() + ".txt"))
                        {
                            //没有则创建这个文件
                            fs = new FileStream("C:\\Users\\hw\\Desktop\\编译原理课程设计\\输出文件\\TransformerRes" +
i.ToString() + ".txt", FileMode.Create, FileAccess.Write);//创建写入文件
                            sw = new StreamWriter(fs);
                            sw.WriteLine();
                            sw.WriteLine();
```

```
                    foreach (var item in TransformerRes)
                    {
                        sw.WriteLine(item);
                    }
                    sw.Close();
                    fs.Close();
                    break;
                }
            }

            lbPath.Text = "保存成功";
        }
        else
        {
            lbPath.Text = "保存失败,请打开需要分析的文件";
        }
    }
```

注：其余代码可参考随报告附加的源程序文件，在运行代码时注意更改文件输入输出的路径。