目录

词法、语法、语义分析

布尔表达式的词法、语法、语义分析程序（C++）

摘要：本系统设计于 2020 的编译原理课设，该系统提供了了半自动与全手动的工作模式，实现了对布尔表达式文法的词法分析，语法分析以及语义分析，并将其中的分析过程与结果根据用户需求保存在本地文件中。之后我也选取了一部分样例进行测试，结果符合预期。最后我也列出了对本系统的改进方案。
关键字：词法、语法、语义、编译原理、C++

Lexical, Grammatical and Semantic Analysis

Lexical, Grammatical and Semantic Analysis Program of Boolean Expression (C++)

Computer Science and Technology Major　Ma Shufan
Tutor　Huang Fen

Abstract: This system is designed in the course of compiler principles in 2020. The system provides semi-automatic and full manual working modes, realizes lexical analysis, syntax analysis and semantic analysis of Boolean expression grammar, and saves the analysis process and results in local files according to user requirements. After that, I also selected some samples for testing, and the results are in line with the expectations. Finally, I also listed the improvement scheme of this system.
Key words: Morphology, grammar, semantics, compiler principles, C++

1　系统简介

该系统启发自本学期的“编译原理”课程，在本系统中，我结合课程所学，用C++语言编程开发了一款可以识别布尔表达式的文法编译器，主要包括输入输出模块，词法分析模块，语法分析模块以及语义分析模块四大模块。

1.1 整体设计思想

系统主要由四大模块组成，分别是输入输出模块，词法分析模块，语法分析

模块以及语义分析模块，各个模块又由多个封装好的函数组成。本系统同时也包括了文件读入，文件保存，词法分析以，语法分析，语义分析以及错误提示的功能。同时本系统也有多种工作模式，批量读取/逐个读取，半自动分析/全手动分析，控制台输出/文件输出三大类工作模式。

1.2 系统的主要功能

1.2.1 文件读入与保存

该系统可以实现读取本地的文件中的代码串，并将其转换为系统识别的字符串进入词法分析程序。同时，该系统也具备将分析过程保存到本地的功能，从而方便用户理解布尔表达式的分析过程。

1.2.2 布尔表达式的分析

该系统可以将读取到的代码串，经过词法分析，语法分析以及语义分析，从而获得表达式的真值，并且向用户展示各个分析的过程。当然，若词法分析或语法分析中出现了错误，则不能进行下一步分析。

1.2.3 错误提示

该系统可以根据各个阶段的分析进行报错处理，并给出相应的可行的解决方案。

1.3 系统的工作模式

1.3.1 文件读取的模式

该类下的模式一：批量读取。该批量是针对代码串（布尔表达式）而言，即该模式下，系统将只读取 codeAll.txt 文件中的内容，该文本文件中可以存放多个布尔表达式，系统识别要求是按照终结符#隔开。

该类下的模式二：逐个读取。系统将读取 code00.txt-code09.txt 十个文本文件，其中每个文本文件中存放的字符串被认为是一个布尔表达式。

1.3.2 分析的模式

该类下的模式一：半自动分析。在该模式下，系统将按照默认设置进行分析。在词法语法分析是遇到错误后，给出错误信息后则进行下一条表达式的分析，并不会退出程序。

该类下的模式二：全手动分析。在该模式下，系统将在每一步分析后暂停，等待用户操作，是继续分析还是结束程序。

1.3.3 输出的模式

该类下的模式一：控制台输出。在该模式下，系统中的所有输出都是指向控制台。

该类下的模式二：文件输出。在该模式下，系统中除了提示用户输出的内容外，所有分析过程都会保存到文件中。

1.4 系统主要结构

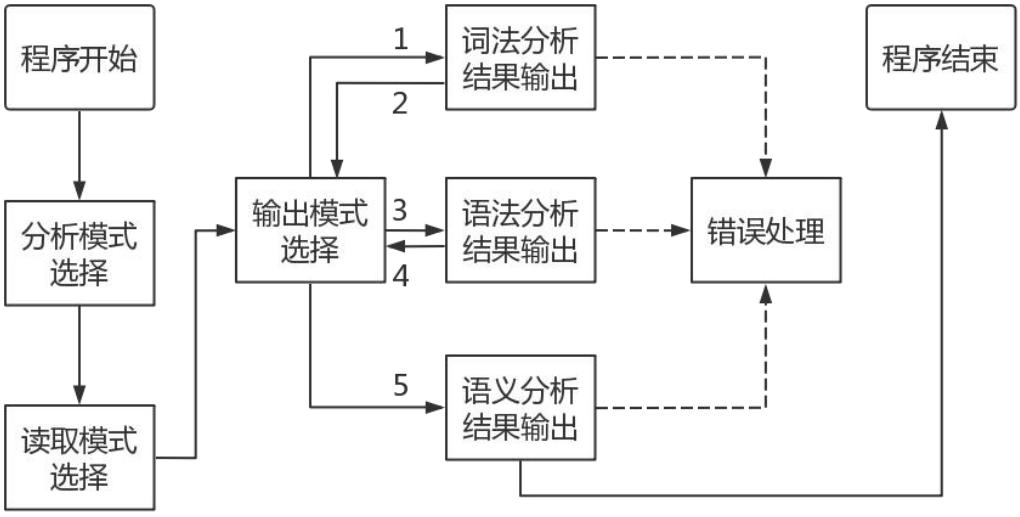


图 1 系统主要结构

2 系统分析以及设计

2.1 适用的文法

布尔表达式的文法表示如下，E 为非终结符，rop 表示比较运算符，id 表示数字或者单个字母。

E -> E or E

E -> E and E

E -> not E

E -> (E)

E -> id rop id

E -> true

E -> false

2.2 词法分析

2.2.1 可识别的单词序列

保留字：or，and，not，false，true；

运算符：<，>，<=，>=，==，!=；

数字：正整数，负整数，正小数，负小数，零既是正整数又是负整数，又是正小数，又是负小数；

字母：a-z 或 A-Z；

界符：(，)；

表达式结束符：#；

其他字符：非法。

2.2.2 词法规则状态转换图

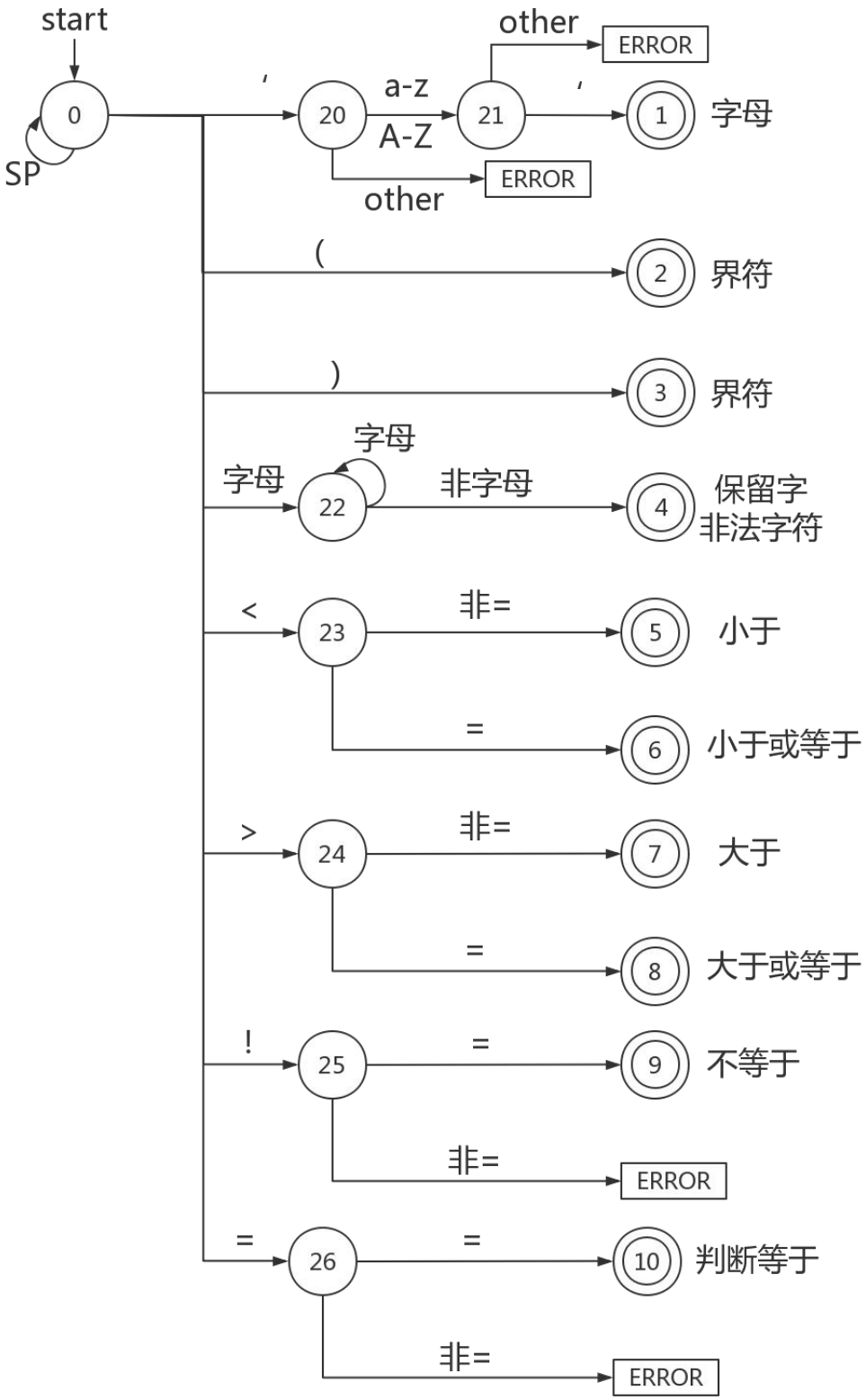


图 2 词法规则状态转换图-1

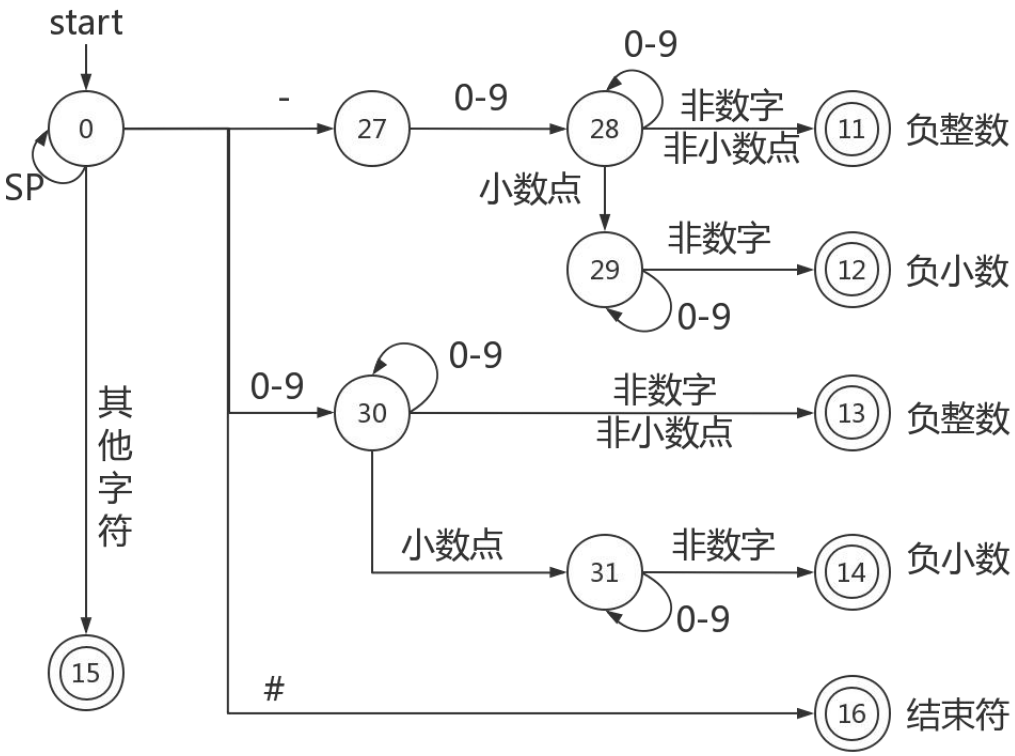


图 3 词法规则状态转换图-2

2.2.3 词法分析模块流程图

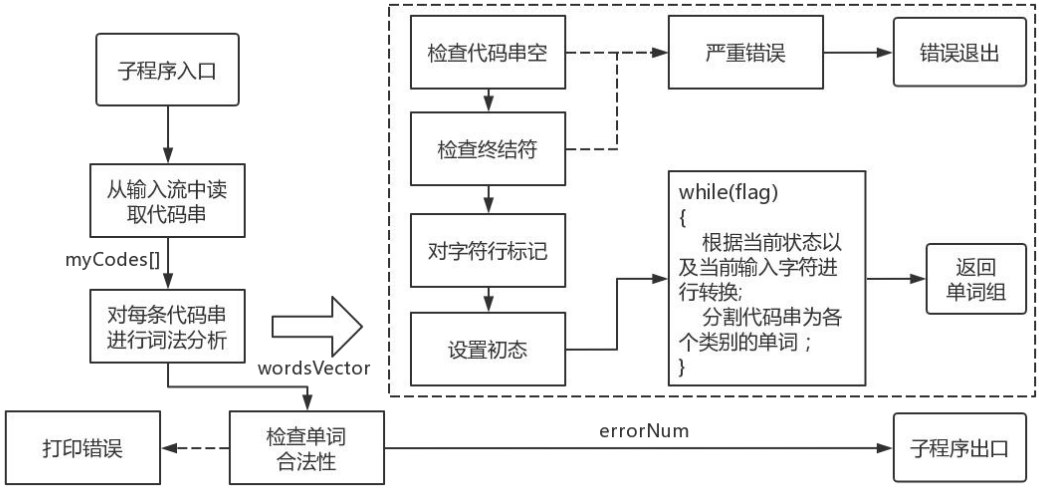


图 4 词法分析程序流程图

2.3 语法分析

2.3.1 拓展后的文法

0：S -> E

1：E -> E or E

2：E -> E and E
3：E -> not E
4：E -> (E)
5：E -> id rop id
6：E -> true
7：E -> false

2.3.2 文法的项目集

[0] S ->·E
[1] S -> E·
[2] E ->·E or E
[3] E -> E·or E
[4] E -> E or·E
[5] E -> E or E·
[6] E ->·E and E
[7] E -> E·and E
[8] E -> E and·E
[9] E -> E and E·
[10] E ->·not E
[11] E -> not·E
[12] E -> not E·
[13] E ->·(E)
[14] E -> (·E)
[15] E -> (E·)
[16] E -> (E)·
[17] E ->·id rop id
[18] E -> id·rop id
[19] E -> id rop·id
[20] E -> id rop id·
[21] E ->·true
[22] E -> true·
[23] E ->·false
[24] E -> false·

2.3.3 项目集规范族

表 1 项目集规范族

序号	规范族所含的项目集
I-0	S -> · E，E -> · E or E，E -> · E and E，E -> · not E， E -> · (E)，E -> · id rop id，E -> · true，E -> · false
I-1	S -> E ·，E -> E · or E，E -> E · and E
I-2	E -> not · E，E -> · E or E，E -> · E and E，E -> · not E E -> · (E)，E -> · id rop id，E -> · true，E -> · false
I-3	E -> (· E)，E -> · E or E，E -> · E and E，E -> · not E E -> · (E)，E -> · id rop id，E -> · true，E -> · false
I-4	E -> id · rop id
I-5	E -> true ·
I-6	E -> false ·
I-7	E -> E or · E，E -> · E or E，E -> · E and E，E -> · not E E -> · (E)，E -> · id rop id，E -> · true，E -> · false
I-8	E -> E and · E，E -> · E or E，E -> · E and E，E -> · not E E -> · (E)，E -> · id rop id，E -> · true，E -> · false
I-9	E -> not E ·，E -> E · or E，E -> E · and E
I-10	E -> (E ·)，E -> E · or E，E -> E · and E
I-11	E -> id rop · id
I-12	E -> id rop id ·
I-13	E -> E or E ·，E -> E · or E，E -> E · and E
I-14	E -> E and E ·，E -> E · or E，E -> E · and E
I-15	E -> (E) ·

2.3.4 识别活前缀的 DFA

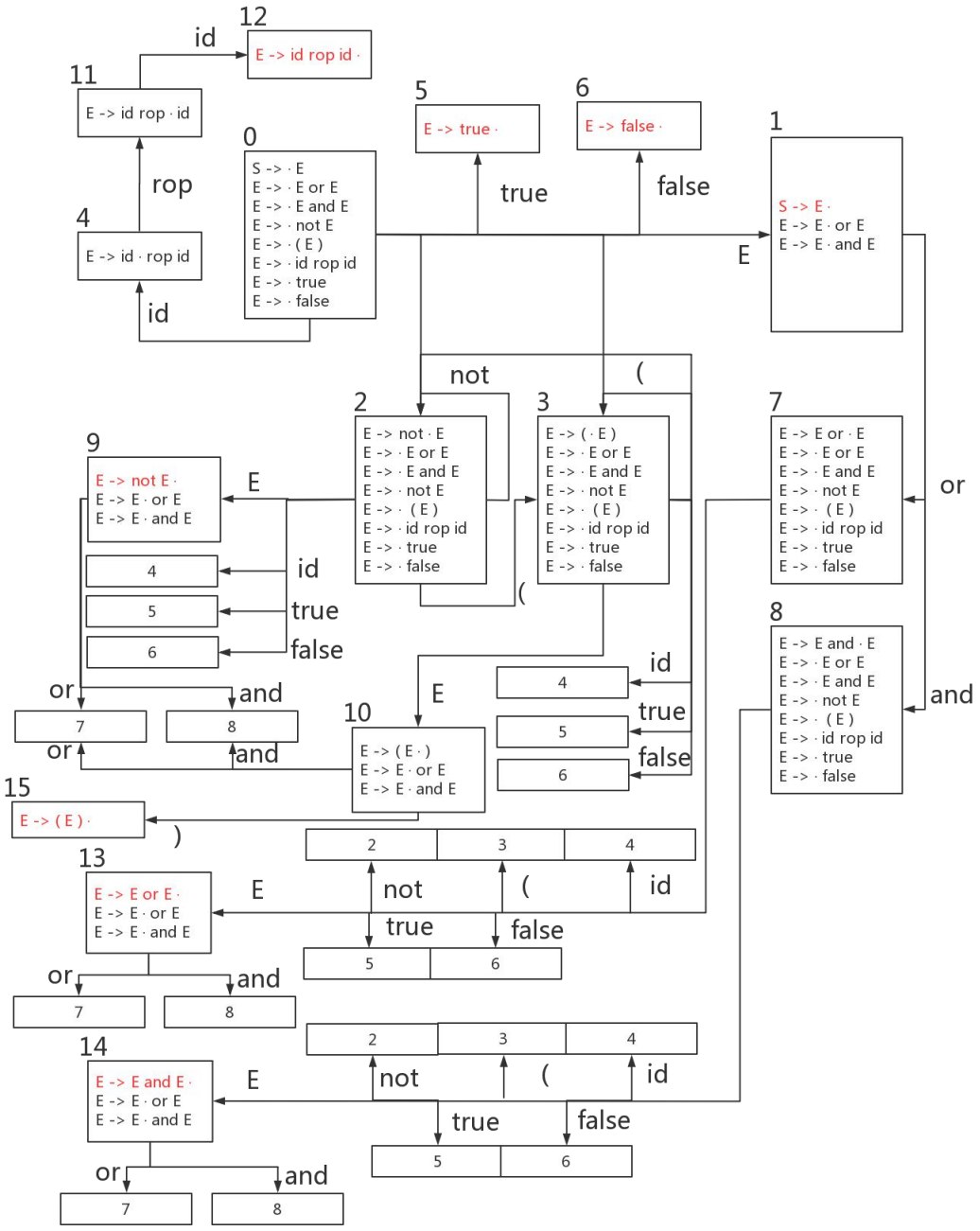


图 5 识别活前缀的 DFA

2.3.5 解决冲突后的 SLR(1)分析表

文法含有的移进-规约冲突 I-1，I-9，I-13 与 I-14。

FIRST(S)={not, (, id, true, false}

FIRST(E)={not, (, id, true, false}

FOLLOW(S)={#}

FOLLOW(E)={#, or, and,)}

运算符的优先级（自低到高）：or，and，not，rop

表 2 SLR(1)分析表

	Action									Goto
	or	and	not	(	)	id	rop	true	false	#	E
0			S2	S3		S4		S5	S6		1
1	S7	S8								acc	
2			S2	S3		S4		S5	S6		9
3			S2	S3		S4		S5	S6		10
4							S11				
5	r6	r6			r6					r6	
6	r7	r7			r7					r7	
7			S2	S3		S4		S5	S6		13
8			S2	S3		S4		S5	S6		14
9	r3	r3			r3					r3	
10	S7	S8			S15						
11						S12					
12	r5	r5			r5					r5	
13	r1	s8			r1					r1	
14	r2	r2			r2					r2	
15	r4	r4			r4					r4	

2.3.6 语法分析模块流程图

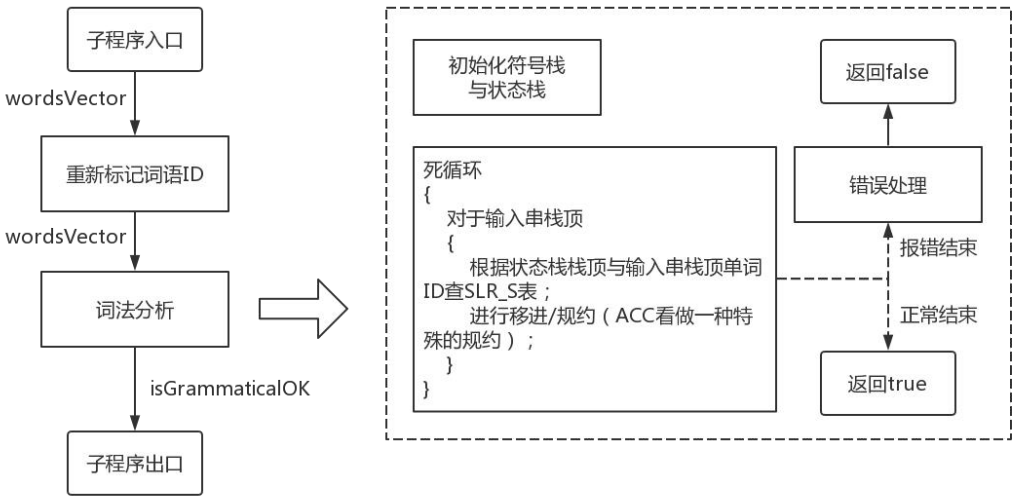


图 6 语法分析模块流程图

2.4 语义分析

2.4.1 逆波兰式

也称为后缀表达式，我们所输入的布尔表达式为中缀表达式，而计算机更方便根据逆波兰式进行计算（先进后出的栈结构），所以我在中间代码生成前首先求出输入的布尔表达式对应的逆波兰式。

2.4.2 中间代码生成

在生成逆波兰式后，系统会根据逆波兰式进行表达式求值，同时将求值过程以四元式的形式输出展示出来。

2.4.3 语义分析模块流程图

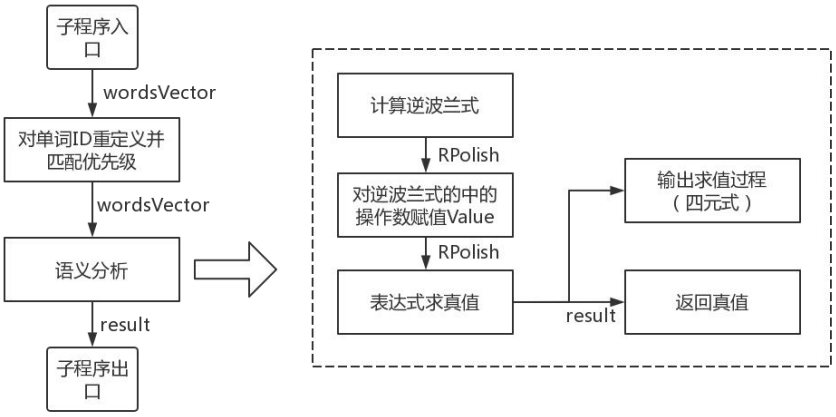


图 7 语义分析模块流程图

2.5 错误处理

本系统总共有两类错误，未知的错误与分析错误，其中未知的错误是程序运行中由于不可控因素发生的错误，从而导致程序无法正常运行；分析错误则为词法分析与语法分析中产生的错误。

在词法分析的错误中，ERROR(0)与 ERROR(1)为严重错误，即输入的布尔表达式为空或者没有结束符。此阶段产生的其他错误代号为 ERROR(1x)的形式。

在语法分析的错误中，此阶段产生的错误代号为 ERROR(2x)与 ERROR(3x)的形式。

3 代码实现

3.1 代码文件架构

头文件包括 head.h，word.h，File.h，AnalysisFunction.h。head.h 文件主要包含系统要用到的全局变量，word.h 文件主要包括对单词类及其方法的声明，File.h 文件主要包括文件输出类以及文件输入输出函数的声明，AnalysisFunction.h 文件责包括分析过程中用到的全部函数声明以及主函数调用子程序的全部声明。而在此之下的 cpp 文件则为对应的实现方法。

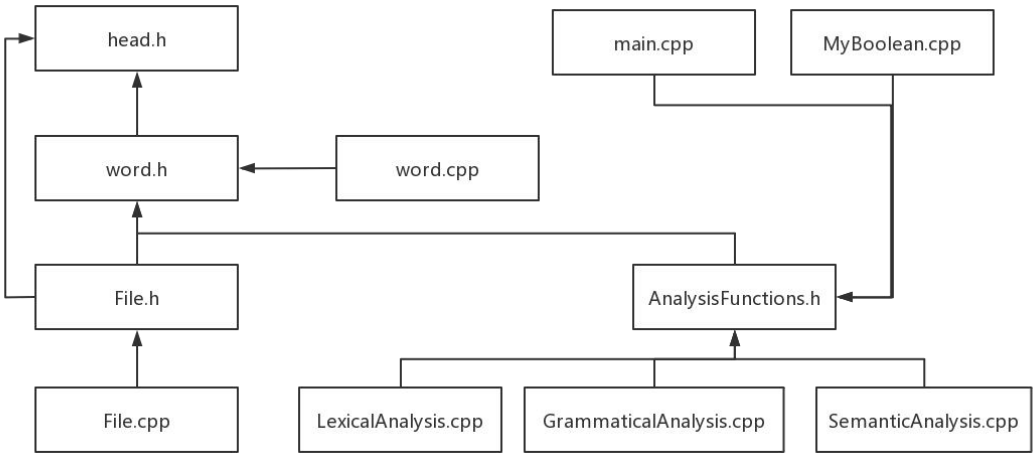


图 8 代码文件架构

3.2 基本数据结构

```
class Word
{
protected:
    // 属性
    int ID;          //记录读入为单词的顺序，在语法分析阶段被重定义为对应的词语表的位置
    int Type;        //一级分类
    int Type2;           //二级分类（仅分类保留字，运算符和数字）
    string Data;     //保存单词中的字符
    int Rows;        // 记录词语在第几行
    double Value; // 用于语义分析，表示操作数 Data 对应的真值，用 0 和 1 表示 true 和 false
    int TempNumber;// 用于语义分析，表示中间代码生成时 temp 变量的编号
public:
    // 方法，这里只列举重要方法，忽略了 set 与 get 方法
    Word();  // 默认构造函数
    Word(int id, int type, int type2, string data, int rows);  //带参构造函数
    Word(const Word &temp);    // 拷贝构造函数
    void addCharToData(char temp);  // 在 Data 后面追加字符
    void doCheck();    // 对单词进行二次分类，并标记非法单词（词法分析）
    void doPrint();// 打印单词信息（词法分析）
    void doPrintDataByID();      // 按照 ID 打印单词（语法分析）
};
```

3.3 全局常量

```
//全部的单词种类
const string WORD_TYPE[] = { "Reserved symbol", "Operator symbol", "Number symbol", "Letter symbol", "Boundary symbol", "Other symbol" , "End symbol"};
const int TYPE_NUM = 6;
//全部的保留字
const string RESERVED_ALL[] = { "or", "and", "not", "true", "false" };
const int RESERVED_NUM = 5;
//全部的运算符
const string OPERATOR_ALL[] = { "<",">","<=",">=","==","!=" };
const int OPERATOR_NUM = 6;
//全部可识别的数字类型
const string NUMBER_ALL[] = { "-Int", "-Double", "+Int", "+Double" };
const int NUMBER_NUM = 4;
//规约表达式的右部的单词数
const int WORD_NUM_IN_STATUTE[] = { 1, 3, 3, 2, 3, 3, 1, 1 };
const string LEFT_WORD_IN_STATUTE[] = { "S", "E", "E", "E", "E", "E", "E", "E" };
//词语表
const string WORD_LIST[] = { "or", "and", "not", "(", ")", "id", "rop", "true", "false", "#", "E", "S" };
//SLR 分析表，横坐标为状态，纵坐标为词语对应的 ID
```

```
// 0 表示 acc，正数表示目标状态，-1 表示无效（error）
const int SLR[16][11] = {
        {-1, -1, 2, 3, -1, 4, -1, 5, 6, -1, 1},
        {7, 8, -1, -1, -1, -1, -1, -1, -1, 0, -1},
        {-1, -1, 2, 3, -1, 4, -1, 5, 6, -1, 9},
        {-1, -1, 2, 3, -1, 4, -1, 5, 6, -1, 10},
        {-1, -1, -1, -1, -1, -1, 11, -1, -1, -1, -1},
        {6, 6, -1, -1, 6, -1, -1, -1, -1, 6, -1},
        {7, 7, -1, -1, 7, -1, -1, -1, -1, 7, -1},
        {-1, -1, 2, 3, -1, 4, -1, 5, 6, -1, 13},
        {-1, -1, 2, 3, -1, 4, -1, 5, 6, -1, 14},
        {3, 3, -1, -1, 3, -1, -1, -1, -1, 3, -1},
        {7, 8, -1, -1, 15, -1, -1, -1, -1, -1, -1},
        {-1, -1, -1, -1, -1, 12, -1, -1, -1, -1, -1},
        {5, 5, -1, -1, 5, -1, -1, -1, -1, 5, -1},
        {1, 8, -1, -1, 1, -1, -1, -1, -1, 1, -1},
        {2, 2, -1, -1, 2, -1, -1, -1, -1, 2, -1},
        {4, 4, -1, -1, 4, -1, -1, -1, -1, 4, -1}
};
// 0 表示移进，1 表示 goto，-1 表示规约
const int SLR_S[16][11] = {
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {-1, -1, 0, 0, -1, 0, 0, 0, 0, -1, 1},
        {-1, -1, 0, 0, -1, 0, 0, 0, 0, -1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {-1, -1, 0, 0, -1, 0, 0, 0, 0, -1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 1},
        {-1, -1, 0, 0, -1, 0, 0, 0, 0, -1, 1},
        {-1, 0, 0, 0, -1, 0, 0, 0, 0, -1, 1},
        {-1, -1, 0, 0, -1, 0, 0, 0, 0, -1, 1},
        {-1, -1, 0, 0, -1, 0, 0, 0, 0, -1, 1}
};
//  读文件时的文件名
const string READ_FILE_NAMES[10] =
{"code00.txt","code01.txt","code02.txt","code03.txt","code04.txt","code05.txt","code06.txt","cod
e07.txt","code08.txt","code09.txt"};
const string OUT_DIR_NAMES[10] =
{"code00\\","code01\\","code02\\","code03\\","code04\\","code05\\","code06\\","code07\\","code0
```

```
8\\","code09\\"};
const int MAX_READ_FILE_NUM = 10;
const int STATE_NUM = 16;
const int WORD_NUM = 11;
// 语义分析时的重定义 ID
// 下标为 ID，值为对应的单词分类
const string WORD_TYPE_SEM[] = { "or", "and", "not", "(", ")", ">", ">=", "<", "<=", "==", "!=",
"id", "true", "false", "#" };
// 语义分析时的优先级
// 下标对应 ID，值对应优先级
const int PRIORITY[] = { 1, 2, 3, 0, 0, 4, 4, 4, 4, 4, 4, 0, 0, 0, 0 };
```

3.4 核心函数及功能

3.4.1 主函数

int main()是程序的主函数，用户在此选择半自动分析模式还是全手动分析模式。

void autoAnalysis()是半自动分析模式的函数。

void handAnalysis()是全手动分析模式的函数。

3.4.2 词法分析中的函数

vector<Word> doLexicalAnalysis(string input, string fname, vector<int> &statesVector)是词法分析中将布尔表达式字符串分隔为一个个可识别单词的函数。

int doFindLexicalErrors(vector<Word> wordsVector, string fname)是词法分析中的错误处理函数，它会扫描分隔出来的每个单词，并逐个分析给出所有错误以及解决方案。

3.4.3 语法分析中的函数

bool doGrammaticalAnalysis(vector<Word> wordsVector, string fname)是语法分析函数，用于语法分析，返回 true 则语法正确，返回 false 则语法错误。

void doPrintErrors(Word, int, string, Word topWord)是在语法分析中遇到错误时，打印错误信息并给出可行的解决方案。

bool doMoveIn(vector<Word> &symbolStack, vector<int> &stateStack, Word word)是语法分析中的移进函数，返回 true 则移进时没有错误，返回 false 则移进时出现错误。

bool doStatute(vector<Word> &symbolStack, vector<int> &stateStack, int exper_id)是语法分析中的规约函数，返回值同移进函数类似。

void doSetAllID(vector<Word> &wordsVector)是语法分析中最先开始的预处理函数，它将重新标记各个单词的 ID，以方便之后的分析。

bool isACC(vector<Word> &symbolStack, Word word)是语法分析中的判断 acc 函数，当遇到 SLR 分析表中对应值为 0 的情况时，会调用此函数。

3.4.4 语义分析中的函数

bool SemanticAnalysis(vector<Word> wordsVector)是语义分析函数。

void setWordInSemantic(vector<Word> &wordsVector)在语义分析前，重新设置单词 ID 以与优先级匹配。

void doCreateRPolish(vector<Word> input, vector<Word> &output)是语义分析中构建逆波兰式的函数。

void setAllValue(vector<Word> &RPolish)是语义分析中给 id，true，false 类单词设置真值的函数，从而用于计算表达式真值。

void printQuaternion(Word op1, Word op2, int tempNums, string p)是语义分析中计算表达式真值时展示计算过程（四元式）的函数。+

bool doCalculation(vector<Word> RPolish)是语义分析中计算表达式真值的函数。

3.5 核心分析算法详解

3.5.1 词法分析算法

```
flag=true;
state=0;  // 设置初态
while(flag)
{
    if(不是终结态)
    {
        tempChar=读取下一个字符;
        charStack.push(tempChar);
    }
    switch(state)
    {
        case 0:
        {
            根据 tempChar 判断接下来的状态变化;
            break;
        }
        case 16:
        {
            flag=false;
            word=’#’;
            wordsVector.push(word);
            break;
        }
        case  终结态代号:
        {
            word=charStack 中元素全部出栈;
            wordsVector.push(word);
            state=0;
            break;
        }
        case  其他状态代号:
        {
            根据 tempChar 判断接下来的状态变化;
            if(tempChar 非法)
            {
                将读取字符串的游标滑到第一个进栈字符之后的一个位置;
```

```
                    word=将第一个进栈字符认作单独的单词（Other word）;
                    wordsVector.push(word);
                    state=0;
                    charStack.clear();
                }
                break;
            }
        }
    }
```

3.5.2 语法分析算法

```
symbolStack.init(); //符号栈初始化
stateStack.init();    //状态栈初始化
hasError=false;            //语法分析是否存在错误
flag=true;
state=0;
while(flag)
{
    // 获取待输入串的队头单词
    tempWord=wordsVector.pop();
    topState; // 获取栈顶状态
    wordID; // 获取当前单词的 ID
    slr_s=SLR_S[topState][wordID]; // 查表获取下一步应该做什么
    if(slr_s==0)    // 移进
    {
        if (!doMoveIn(symbolStack, stateStack, tempWord))
        {
                hasError=true;
                flag=false;
                报错;
        }
    }
    else if(slr_s==-1)    // 规约
    {
        // 查 SLR 表看当前是 acc 还是一般规约
        if (SLR[topState][wordID] == -1)
        {
                hasError=true;
                flag=false;
                报错;
        }
        else if (SLR[topState][wordID] == 0)
        {
            // acc
            if (isACC(symbolStack, tempWord))
```

```
                {
                    flag=false;
                    hasError=false;
                }
                else
                {
                    flag=false;
                    hasError=true;
                    报错;
                }
            }
            else
            {
                if(!doStatute(symbolStack,stateStack,SLR[topState][wordID]))
                {
                    hasError=true;
                    flag=false;
                    报错;
                }
            }
        }
        else
        {
            报错;
            flag=false;
            hasError=true;
        }
}
return hasError;
```

3.5.3 语义分析算法

```
// 创建逆波兰式
for(从左往右扫描中缀表达式串 s)
{
    对于每一个操作数或操作符，执行以下操作
    if(扫描到的 s[i]是操作数 DATA)
        将 s[i]添加到输出串中;
    if(扫描到的 s[i]是开括号'(')
        将 s[i]压栈;
    while(扫描到的 s[i]是操作符 OP)
    {
        if(栈为空或栈顶为'('或扫描到的操作符优先级比栈顶操作符高)
        {
            将 s[i]压栈;
            break;
```

```
            }
            else
                出栈至输出串中;
        }
        while(扫描到的 s[i]是闭括号')')
        {
            if(直到遇到开括号'(')
            {
                开括号'('出栈并丢弃;
                break;
            }
            栈中运算符逐个出栈并输出;
        }
}
while(扫描结束而栈中还有操作符)
{
    操作符出栈并加到输出串中;
}
// 表达式求值
for(从左到右扫描逆波兰式中的每个单词)
{
    if(是运算符)
    {
        出栈相应数目的单词;
        temp=运算结果;
        输出当前进行运算的四元式;
        temp 进栈;
    }
    else if(是操作数)
    {
        进栈;
    }
    else
    {
        报错;
    }
}
```

4 测试以及结果

4.1 测试时选择的配置

系统的分析模式：全手动分析模式

文件读取的模式：批量读取 codeAll.txt 文件

结果输出的模式：本地文件保存到 output\codeAll 文件夹下

4.2 测试样例

codeAll.txt 文件中的内容如下所示：

abs (("A" and 'a') flase or -3.21.3 !>= 7)#
(1 != 'a') and op#
(false or (1!=2))and 'j'#
or 2.3 and 'a'#
((false or true)#
not (4 or 'a')#
(1!=-2.32)and(not 3>='a' or false)#
(false or(64 == 'a')) and not true#
(1>=2.3) and ('a'!=3) or true#

4.3 理论结果

将 codeAll.txt 中的布尔表达式按照序号排列为 0-12 号，则理论上的分析如下：

词法错误：0,1；

语法错误：2,3,4,5；

无错误情况下，语义分析结果：

6：true
7：false
8：true

测试的部分布尔表达式的 SLR 分析过程如下：

表 3 ((false or true)#表达式语法分析过程

序号	状态栈	符号栈	待输入串	Action	Goto
0	0	$	((false or true)#	S3	
1	03	$(	(false or true)#	S3	
2	033	$((	false or true)#	S6	
3	0336	$((false	or true)#	r7	10
4	033(10)	$((E	or true)#	S7	
5	033(10)7	$((E or	true)#	S5	
6	033(10)75	$((E or true	)#	r6	13
7	033(10)7(13)	$((E or E	)#	r1	10
8	033(10)	$((E	)#	S15	
9	033(10)(15)	$((E)	#	r4	10
10	03(10)	$(E	#	出错	

表 4 (false or(64 == 'a'))and not true#表达式语法分析过程

	状态栈	符号栈	待输入串	A	G
0	0	$	(false or(id rop id))and not true#	S3	
1	03	$(	false or(id rop id))and not true#	S6	
2	036	$(false	or(id rop id))and not true#	r7	10
3	036(10)	$(E or	or(id rop id))and not true#	S7	
4	036(10)7	$(E or	(id rop id))and not true#	S3	
5	036(10)73	$(E or(	id rop id))and not true#	S4	
6	036(10)734	$(E or(id	rop id))and not true#	S11	

7	036(10)734(11)	$(E or(id rop	id))and	not	true#	S12	
8	036(10)734(11)(12)	$(E or(id rop id	))and	not	true#	r5	10
9	03(10)73(10)	$(E or(E	))and	not	true#	S15	
10	03(10)73(10)(15)	$(E or(E)	)and	not	true#	r4	13
11	03(10)7(13)	$(E or E	)and	not	true#	r1	10
12	03(10)	$(E	)and	not	true#	S15	
13	03(10)(15)	$(E)	and	not	true#	r4	1
14	01	$E	and	not	true#	S8	
15	018	$E and		not	true#	S2	
16	0182	$E and not			true#	S5	
17	01825	$E and not true			#	r6	9
18	01829	$E and not E			#	r3	14
19	018(14)	$E and E			#	r2	1
20	01	$E			#	acc	

4.4 测试结果

文件读取后的结果显示：

```
Read file<F:\学习\桌面实验习题文件夹\编译原理课设\MyBoolean\MyBoolean\input\codeAll.txt> completed.
■■■ 文件: code00.txt ■■■
abs (("A" and 'a') flase or -3.21.3 !>= 7)#
■■■ 文件: code01.txt ■■■

(1 != 'a') and op#
■■■ 文件: code02.txt ■■■

(false or (1!=2))and 'j'#
■■■ 文件: code03.txt ■■■

or 2.3 and 'a'#
■■■ 文件: code04.txt ■■■

(( false or true)#
■■■ 文件: code05.txt ■■■

not (4 or 'a')#
■■■ 文件: code06.txt ■■■

(1!=-2.32)and(not 3>='a' or false)#
■■■ 文件: code07.txt ■■■

(false or(64 == 'a')   ) and not true#
■■■ 文件: code08.txt ■■■

(1>=2.3) and ('a'!=3) or true#
■■■ 文件: code09.txt ■■■

not true#
■■■ 文件: code010.txt ■■■

false#
■■■ 文件: code011.txt ■■■

2 != 3#
■■■ 文件: code012.txt ■■■

not((1==1) or (2!=3) and ('a'>='b')or true)#
以上为读入的全部代码串，共 13组
```

打印 SLR 分析表：

SLR(1)分析表

state	or	and	not	(	)	id	rop	true	false	#	E
0			S2	S3		S4		S5	S6		1
1	S7	S8								acc	
2			S2	S3		S4		S5	S6		9
3			S2	S3		S4		S5	S6		10
4							S11				
5	r6	r6			r6					r6	
6	r7	r7			r7					r7	
7			S2	S3		S4		S5	S6		13
8			S2	S3		S4		S5	S6		14
9	r3	r3			r3					r3	
10	S7	S8			S15						
11						S12					
12	r5	r5			r5					r5	
13	r1	S8			r1					r1	
14	r2	r2			r2					r2	
15	r4	r4			r4					r4	

各个表达式的分析结果

（1）abs (("A" and 'a') flase or -3.21.3 !>= 7)#的词法分析

```
-* 词法分析 *- 开始            文件: code00.txt
-* 词法分析 *- 状态转换过程    文件: code00.txt
0       22      22      22      4       0       0       2       0
2
0       15      0       22      4       0       15      0       0
22
22      22      4       0       0       20      21      1       0
3
0       0       22      22      22      22      22      4       0
0
22      22      4       0       0       27      28      29      29
29
12      0       15      0       30      13      0       0       25
0
24      8       0       0       30      13      0       3       0
16

-* 词法分析 *- 完成   文件: code00.txt
序号    一级分类            二级分类    行标记      字符(串)
0       Other symbol                    None            0       abs
1       Boundary symbol                 None            0       (
2       Boundary symbol                 None            0       (
3       Other symbol                    None            0       "
4       Other symbol                    None            0       A
5       Other symbol                    None            0       "
6       Reserved symbol                 and             0       and
7       Letter symbol                   None            0       a
8       Boundary symbol                 None            0       )
9       Other symbol                    None            0       flase
10      Reserved symbol                 or              0       or
11      Number symbol                   -Double         0       -3.21
12      Other symbol                    None            0       .
13      Number symbol                   +Int            0       3
```

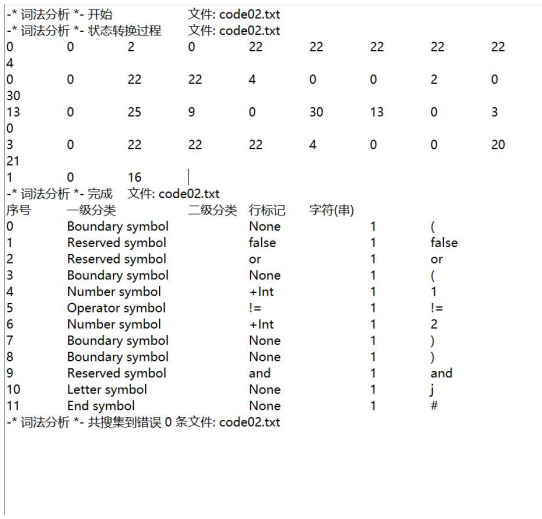
```
2       Boundary symbol         None            0       (
3       Other symbol            None            0       "
4       Other symbol            None            0       A
5       Other symbol            None            0       "
6       Reserved symbol         and             0       and
7       Letter symbol           None            0       a
8       Boundary symbol         None            0       )
9       Other symbol            None            0       flase
10      Reserved symbol         or              0       or
11      Number symbol           -Double         0       -3.21
12      Other symbol            None            0       .
13      Number symbol           +Int            0       3
14      Other symbol            None            0       !
14      Operator symbol         >=              0       >=
15      Number symbol           +Int            0       7
16      Boundary symbol         None            0       )
17      End symbol              None            0       #
ERROR(16): 无法识别的保留字 abs      文件: code00.txt    行: 0
    可能的修改方法: 修改为正确的保留字
ERROR(17): 无法识别的字符 *    文件: code00.txt    行: 0
    可能的修改方法: 尝试删除或替换
ERROR(17): 无法识别的字符 A    文件: code00.txt    行: 0
    可能的修改方法: 尝试删除或替换
ERROR(17): 无法识别的字符 *    文件: code00.txt    行: 0
    可能的修改方法: 尝试删除或替换
ERROR(16): 无法识别的保留字 flase    文件: code00.txt    行: 0
    可能的修改方法: 修改为正确的保留字
ERROR(11): 无法识别的字符 .    文件: code00.txt    行: 0
    可能的修改方法: 尝试小数中只含有一个小数点
ERROR(13): 无法识别的字符 !    文件: code00.txt    行: 0
    可能的修改方法: 尝试将 ! 替换为 !=
-* 词法分析 *- 共搜集到错误 7 条文件: code00.txt
```

（2）(1 != 'a') and op#的词法分析

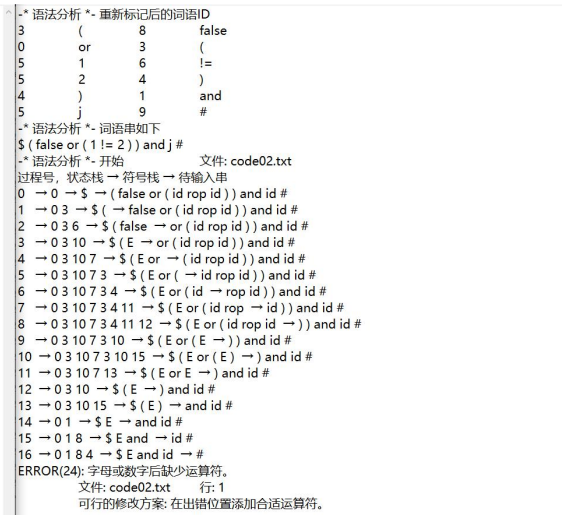
```
-* 词法分析 *- 开始            文件: code01.txt
-* 词法分析 *- 状态转换过程    文件: code01.txt
0       0       2       0       30      13      0       0       25      9
0       0       20      21      1       0       3       0       0       22
22      22      4       0       0       22      22      4       0       16

-* 词法分析 *- 完成   文件: code01.txt
序号    一级分类            二级分类    行标记      字符(串)
0       Boundary symbol                 None            1       (
1       Number symbol                   +Int            1       1
2       Operator symbol                 !=              1       !=
3       Letter symbol                   None            1       a
4       Boundary symbol                 None            1       )
5       Reserved symbol                 and             1       and
6       Other symbol                    None            1       op
7       End symbol                      None            1       #
ERROR(16): 无法识别的保留字 op       文件: code01.txt    行: 1
    可能的修改方法: 修改为正确的保留字
-* 词法分析 *- 共搜集到错误 1 条文件: code01.txt
```

（3）(false or (1!=2))and 'j'#的词法分析与语法分析

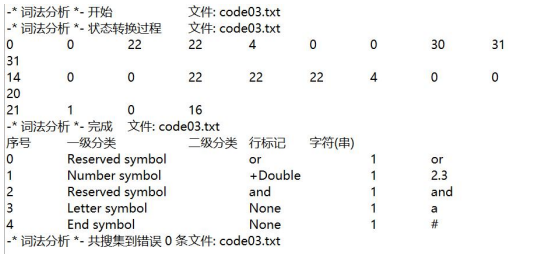


```
-* 词法分析 *- 开始                文件: code02.txt
-* 词法分析 *- 状态转换过程        文件: code02.txt
0       0       2       0       22      22      22      22      22
4
0       0       22      22      4       0       0       2       0
30
13      0       25      9       0       30      13      0       3
0
3       0       22      22      22      4       0       0       20
21
1       0       16
-* 词法分析 *- 完成    文件: code02.txt
序号    一级分类                二级分类  行标记    字符(串)
0       Boundary symbol                 None            1       (
1       Reserved symbol                 false           1       false
2       Reserved symbol                 or              1       or
3       Boundary symbol                 None            1       (
4       Number symbol                   +Int            1       1
5       Operator symbol                 !=              1       !=
6       Number symbol                   +Int            1       2
7       Boundary symbol                 None            1       )
8       Boundary symbol                 None            1       )
9       Reserved symbol                 and             1       and
10      Letter symbol                   None            1       j
11      End symbol                      None            1       #
-* 词法分析 *- 共搜集到错误 0 条文件: code02.txt
```

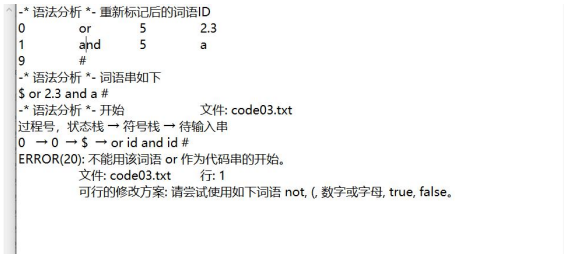


```
-* 语法分析 *- 重新标记后的词语ID
3       (       8       false
0       or      3       (
5       1       6       !=
5       2       4       )
4       )       1       and
5       j       9       #
-* 语法分析 *- 词语串如下
$ ( false or ( 1 != 2 ) ) and j #
-* 语法分析 *- 开始                文件: code02.txt
过程号，状态栈 → 符号栈 → 待输入串
0  → 0  → $  → ( false or ( id rop id ) ) and id #
1  → 0 3  → $ (  → false or ( id rop id ) ) and id #
2  → 0 3 6  → $ ( false  → or ( id rop id ) ) and id #
3  → 0 3 10  → $ ( E  → or ( id rop id ) ) and id #
4  → 0 3 10 7  → $ ( E or  → ( id rop id ) ) and id #
5  → 0 3 10 7 3  → $ ( E or (  → id rop id ) ) and id #
6  → 0 3 10 7 3 4  → $ ( E or ( id  → rop id ) ) and id #
7  → 0 3 10 7 3 4 11  → $ ( E or ( id rop  → id ) ) and id #
8  → 0 3 10 7 3 4 11 12  → $ ( E or ( id rop id  → ) ) and id #
9  → 0 3 10 7 3 10  → $ ( E or ( E  → ) ) and id #
10  → 0 3 10 7 3 10 15  → $ ( E or ( E )  → ) and id #
11  → 0 3 10 7 13  → $ ( E or E  → ) and id #
12  → 0 3 10  → $ ( E  → ) and id #
13  → 0 3 10 15  → $ ( E )  → and id #
14  → 0 1  → $ E  → and id #
15  → 0 1 8  → $ E and  → id #
16  → 0 1 8 4  → $ E and id  → #
ERROR(24): 字母或数字后缺少运算符。
        文件: code02.txt        行: 1
        可行的修改方案: 在出错位置添加合适运算符。
```

（4）or 2.3 and 'a'#的词法分析与语法分析

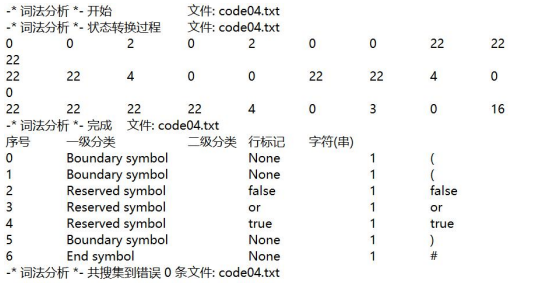


```
-* 词法分析 *- 开始                文件: code03.txt
-* 词法分析 *- 状态转换过程        文件: code03.txt
0       0       22      22      4       0       0       30      31
31
14      0       0       22      22      22      4       0       0
20
21      1       0       16
-* 词法分析 *- 完成    文件: code03.txt
序号    一级分类                二级分类  行标记    字符(串)
0       Reserved symbol                 or              1       or
1       Number symbol                   +Double         1       2.3
2       Reserved symbol                 and             1       and
3       Letter symbol                   None            1       a
4       End symbol                      None            1       #
-* 词法分析 *- 共搜集到错误 0 条文件: code03.txt
```

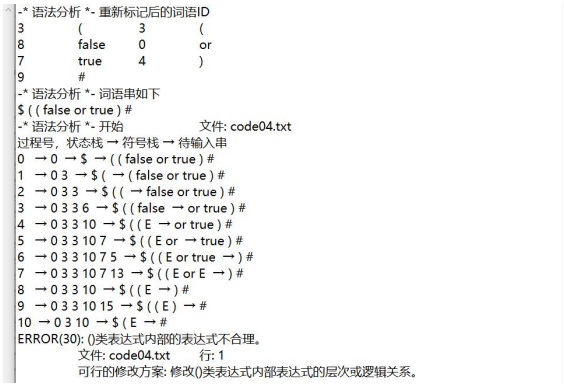


```
-* 语法分析 *- 重新标记后的词语ID
0       or      5       2.3
1       and     5       a
9       #
-* 语法分析 *- 词语串如下
$ or 2.3 and a #
-* 语法分析 *- 开始                文件: code03.txt
过程号，状态栈 → 符号栈 → 待输入串
0  → 0  → $  → or id and id #
ERROR(20): 不能用该词语 or 作为代码串的开始。
        文件: code03.txt        行: 1
        可行的修改方案: 请尝试使用如下词语 not, (, 数字或字母, true, false。
```

（5）((false or true)#的词法分析与语法分析

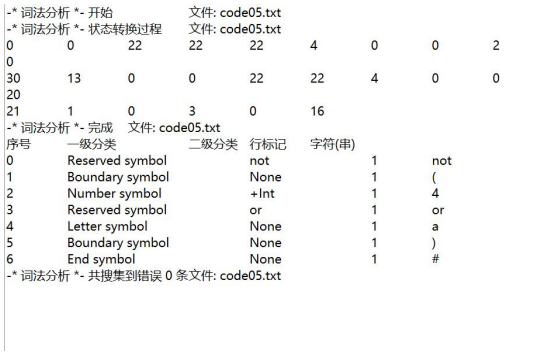


```
-* 词法分析 *- 开始                文件: code04.txt
-* 词法分析 *- 状态转换过程        文件: code04.txt
0       0       2       0       2       0       0       22      22
22
22      22      4       0       0       22      22      4       0
0
22      22      22      22      4       0       3       0       16
-* 词法分析 *- 完成    文件: code04.txt
序号    一级分类                二级分类  行标记    字符(串)
0       Boundary symbol                 None            1       (
1       Boundary symbol                 None            1       (
2       Reserved symbol                 false           1       false
3       Reserved symbol                 or              1       or
4       Reserved symbol                 true            1       true
5       Boundary symbol                 None            1       )
6       End symbol                      None            1       #
-* 词法分析 *- 共搜集到错误 0 条文件: code04.txt
```

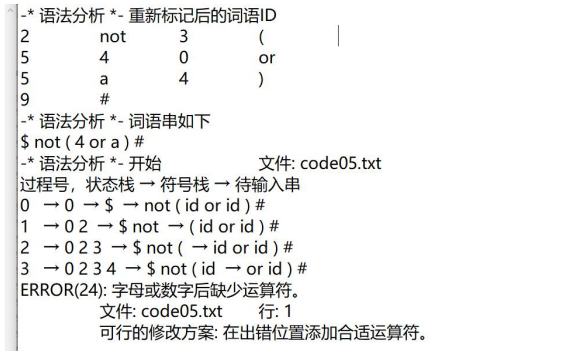


```
-* 语法分析 *- 重新标记后的词语ID
3       (       3       (
8       false   0       or
7       true    4       )
9       #
-* 语法分析 *- 词语串如下
$ ( ( false or true ) #
-* 语法分析 *- 开始                文件: code04.txt
过程号，状态栈 → 符号栈 → 待输入串
0  → 0  → $  → ( ( false or true ) #
1  → 0 3  → $ (  → ( false or true ) #
2  → 0 3 3  → $ ( (  → false or true ) #
3  → 0 3 3 6  → $ ( ( false  → or true ) #
4  → 0 3 3 10  → $ ( ( E  → or true ) #
5  → 0 3 3 10 7  → $ ( ( E or  → true ) #
6  → 0 3 3 10 7 5  → $ ( ( E or true  → ) #
7  → 0 3 3 10 7 13  → $ ( ( E or E  → ) #
8  → 0 3 3 10  → $ ( ( E  → ) #
9  → 0 3 3 10 15  → $ ( ( E )  → #
10  → 0 3 10  → $ ( E  → #
ERROR(30): ()类表达式内部的表达式不合理。
        文件: code04.txt        行: 1
        可行的修改方案: 修改()类表达式内部表达式的层次或逻辑关系。
```

（6）not (4 or 'a')#的词法分析与语法分析



```
-* 词法分析 *- 开始                文件: code05.txt
-* 词法分析 *- 状态转换过程        文件: code05.txt
0       0       22      22      22      4       0       0       2
0
30      13      0       0       22      22      4       0       0
20
21      1       0       3       0       16
-* 词法分析 *- 完成    文件: code05.txt
序号    一级分类                二级分类  行标记    字符(串)
0       Reserved symbol                 not             1       not
1       Boundary symbol                 None            1       (
2       Number symbol                   +Int            1       4
3       Reserved symbol                 or              1       or
4       Letter symbol                   None            1       a
5       Boundary symbol                 None            1       )
6       End symbol                      None            1       #
-* 词法分析 *- 共搜集到错误 0 条文件: code05.txt
```



```
-* 语法分析 *- 重新标记后的词语ID
2       not     3       (
5       4       0       or
5       a       4       )
9       #
-* 语法分析 *- 词语串如下
$ not ( 4 or a ) #
-* 语法分析 *- 开始                文件: code05.txt
过程号，状态栈 → 符号栈 → 待输入串
0  → 0  → $  → not ( id or id ) #
1  → 0 2  → $ not  → ( id or id ) #
2  → 0 2 3  → $ not (  → id or id ) #
3  → 0 2 3 4  → $ not ( id  → or id ) #
ERROR(24): 字母或数字后缺少运算符。
        文件: code05.txt        行: 1
        可行的修改方案: 在出错位置添加合适运算符。
```

（7）(1!=-2.32)and(not 3>='a' or false)#的全部分析过程

```
-* 词法分析 *- 开始              文件: code06.txt
-* 词法分析 *- 状态转换过程      文件: code06.txt
0       0       2       0       30      13      0       25      9       0
27      28      29      29      29      12      0       3       0       22
22      22      4       0       2       0       22      22      22      4
0       0       30      13      0       24      8       0       20      21
1       0       0       22      22      4       0       0       22      22
22      22      22      4       0       3       0       16
-* 词法分析 *- 完成    文件: code06.txt
序号    一级分类                二级分类  行标记      字符(串)
0       Boundary symbol                 None            1       (
1       Number symbol                   +Int            1       1
2       Operator symbol                 !=              1       !=
3       Number symbol                   -Double         1       -2.32
4       Boundary symbol                 None            1       )
5       Reserved symbol                 and             1       and
6       Boundary symbol                 None            1       (
7       Reserved symbol                 not             1       not
8       Number symbol                   +Int            1       3
9       Operator symbol                 >=              1       >=
10      Letter symbol                   None            1       a
11      Reserved symbol                 or              1       or
12      Reserved symbol                 false           1       false
13      Boundary symbol                 None            1       )
14      End symbol                      None            1       #
-* 词法分析 *- 共搜集到错误 0 条文件: code06.txt
```

过程号，状态栈 → 符号栈 → 待输入串
0 → 0 → $ → (id rop id) and (not id rop id or false) #
1 → 0 3 → $ (→ id rop id) and (not id rop id or false) #
2 → 0 3 4 → $ (id → rop id) and (not id rop id or false) #
3 → 0 3 4 11 → $ (id rop → id) and (not id rop id or false) #
4 → 0 3 4 11 12 → $ (id rop id →) and (not id rop id or false) #
5 → 0 3 10 → $ (E →) and (not id rop id or false) #
6 → 0 3 10 15 → $ (E) → and (not id rop id or false) #
7 → 0 1 → $ E → and (not id rop id or false) #
8 → 0 1 8 → $ E and → (not id rop id or false) #
9 → 0 1 8 3 → $ E and (→ not id rop id or false) #
10 → 0 1 8 3 2 → $ E and (not → id rop id or false) #
11 → 0 1 8 3 2 4 → $ E and (not id → rop id or false) #
12 → 0 1 8 3 2 4 11 → $ E and (not id rop → id or false) #
13 → 0 1 8 3 2 4 11 12 → $ E and (not id rop id → or false) #
14 → 0 1 8 3 2 9 → $ E and (not E → or false) #
15 → 0 1 8 3 10 → $ E and (E → or false) #
16 → 0 1 8 3 10 7 → $ E and (E or → false) #
17 → 0 1 8 3 10 7 6 → $ E and (E or false →) #
18 → 0 1 8 3 10 7 13 → $ E and (E or E →) #
19 → 0 1 8 3 10 → $ E and (E →) #
20 → 0 1 8 3 10 15 → $ E and (E) → #
21 → 0 1 8 14 → $ E and E → #
22 → 0 1 → $ E → #
acc
☆ 语法分析没有错误

-* 语义分析 *-　　生成中间代码（逆波兰式）
1 -2.32 != 3 a >= not false or and
-* 语义分析 *- 给逆波兰式中的操作数赋真值
1 → 1
-2.32 → -2.32
3 → 3
a → 97
false → 0
-* 语义分析 *-　　中简代码生成
(!= 1 -2.32 T1)
(>= 3 a T2)
(not T2 _ T3)
(or T3 false T4)
(and T1 T4 T5)
-* 语义分析 *-　　完成
布尔表达式值 = true

（8）(false or(64 == 'a')　　) and not true#的全部分析过程

-* 词法分析 *- 开始　　文件: code07.txt
-* 词法分析 *- 状态转换过程　　文件: code07.txt

0	0	2	0	22	22	22	22	22	4
0	0	22	22	4	0	2	0	30	30
13	0	0	26	10	0	0	20	21	1
0	3	0	0	0	0	3	0	0	22
22	22	4	0	0	22	22	22	4	0
0	22	22	22	22	4	0	16		

-* 词法分析 *- 完成　文件: code07.txt

序号	一级分类	二级分类	行标记	字符(串)	
0	Boundary symbol		None	1	(
1	Reserved symbol		false	1	false
2	Reserved symbol		or	1	or
3	Boundary symbol		None	1	(
4	Number symbol		+Int	1	64
5	Operator symbol		==	1	==
6	Letter symbol		None	1	a
7	Boundary symbol		None	1	)
8	Boundary symbol		None	1	)
9	Reserved symbol		and	1	and
10	Reserved symbol		not	1	not
11	Reserved symbol		true	1	true
12	End symbol		None	1	#

-* 词法分析 *- 共搜集到错误 0 条文件: code07.txt

```
-* 语法分析 *- 词语串如下
$ ( false or ( 64 == a ) ) and not true #
-* 语法分析 *- 开始                    文件: code07.txt
过程号，状态栈 → 符号栈 → 待输入串
0  → 0  → $  → ( false or ( id rop id ) ) and not true #
1  → 0 3  → $ (  → false or ( id rop id ) ) and not true #
2  → 0 3 6  → $ ( false  → or ( id rop id ) ) and not true #
3  → 0 3 10  → $ ( E  → or ( id rop id ) ) and not true #
4  → 0 3 10 7  → $ ( E or  → ( id rop id ) ) and not true #
5  → 0 3 10 7 3  → $ ( E or (  → id rop id ) ) and not true #
6  → 0 3 10 7 3 4  → $ ( E or ( id  → rop id ) ) and not true #
7  → 0 3 10 7 3 4 11  → $ ( E or ( id rop  → id ) ) and not true #
8  → 0 3 10 7 3 4 11 12  → $ ( E or ( id rop id  → ) ) and not true #
9  → 0 3 10 7 3 10  → $ ( E or ( E  → ) ) and not true #
10  → 0 3 10 7 3 10 15  → $ ( E or ( E )  → ) and not true #
11  → 0 3 10 7 13  → $ ( E or E  → ) and not true #
12  → 0 3 10  → $ ( E  → ) and not true #
13  → 0 3 10 15  → $ ( E )  → and not true #
14  → 0 1  → $ E  → and not true #
15  → 0 1 8  → $ E and  → not true #
16  → 0 1 8 2  → $ E and not  → true #
17  → 0 1 8 2 5  → $ E and not true  → #
18  → 0 1 8 2 9  → $ E and not E  → #
19  → 0 1 8 14  → $ E and E  → #
20  → 0 1  → $ E  → #
acc
☆ 语法分析没有错误
```

```
-* 语义分析 *-          生成中间代码（逆波兰式）
false 64 a == or true not and
-* 语义分析 *- 给逆波兰式中的操作数赋真值
false → 0
64 → 64
a → 97
true → 1
-* 语义分析 *-          中简代码生成
( == 64 a T1 )
( or false T1 T2 )
( not true __ T3 )
( and T2 T3 T4 )
-* 语义分析 *-          完成
布尔表达式值 = false
```

（9）(1>=2.3) and ('a'!=3) or true#的全部分析过程

```
-* 词法分析 *- 开始                文件: code08.txt
-* 词法分析 *- 状态转换过程        文件: code08.txt
0       0       2       0       30      13      0       24      8       0
30      31      31      14      0       3       0       0       22      22
22      4       0       0       2       0       20      21      1       0
25      9       0       30      13      0       3       0       0       22
22      4       0       0       22      22      22      22      4       0
16
-* 词法分析 *- 完成    文件: code08.txt
序号    一级分类                二级分类  行标记      字符(串)
0       Boundary symbol                 None            1       (
1       Number symbol                   +Int            1       1
2       Operator symbol                 >=              1       >=
3       Number symbol                   +Double         1       2.3
4       Boundary symbol                 None            1       )
5       Reserved symbol                 and             1       and
6       Boundary symbol                 None            1       (
7       Letter symbol                   None            1       a
8       Operator symbol                 !=              1       !=
9       Number symbol                   +Int            1       3
10      Boundary symbol                 None            1       )
11      Reserved symbol                 or              1       or
12      Reserved symbol                 true            1       true
13      End symbol                      None            1       #
-* 词法分析 *- 共搜集到错误 0 条文件: code08.txt

-* 语法分析 *- 词语串如下
$ ( 1 >= 2.3 ) and ( a != 3 ) or true #
-* 语法分析 *- 开始                文件: code08.txt
过程号，状态栈 → 符号栈 → 待输入串
0  → 0  → $  → ( id rop id ) and ( id rop id ) or true #
1  → 0 3  → $ (  → id rop id ) and ( id rop id ) or true #
2  → 0 3 4  → $ ( id  → rop id ) and ( id rop id ) or true #
3  → 0 3 4 11  → $ ( id rop  → id ) and ( id rop id ) or true #
4  → 0 3 4 11 12  → $ ( id rop id  → ) and ( id rop id ) or true #
5  → 0 3 10  → $ ( E  → ) and ( id rop id ) or true #
6  → 0 3 10 15  → $ ( E )  → and ( id rop id ) or true #
7  → 0 1  → $ E  → and ( id rop id ) or true #
8  → 0 1 8  → $ E and  → ( id rop id ) or true #
9  → 0 1 8 3  → $ E and (  → id rop id ) or true #
10  → 0 1 8 3 4  → $ E and ( id  → rop id ) or true #
11  → 0 1 8 3 4 11  → $ E and ( id rop  → id ) or true #
12  → 0 1 8 3 4 11 12  → $ E and ( id rop id  → ) or true #
13  → 0 1 8 3 10  → $ E and ( E  → ) or true #
14  → 0 1 8 3 10 15  → $ E and ( E )  → or true #
15  → 0 1 8 14  → $ E and E  → or true #
16  → 0 1  → $ E  → or true #
17  → 0 1 7  → $ E or  → true #
18  → 0 1 7 5  → $ E or true  → #
19  → 0 1 7 13  → $ E or E  → #
20  → 0 1  → $ E  → #
acc
☆ 语法分析没有错误
```

```
-* 语义分析 *-          生成中间代码（逆波兰式）
1 2.3 >= a 3 != and true or
-* 语义分析 *- 给逆波兰式中的操作数赋真值
1 → 1
2.3 → 2.3
a → 97
3 → 3
true → 1
-* 语义分析 *-          中简代码生成
( >= 1 2.3 T1 )
( != a 3 T2 )
( and T1 T2 T3 )
( or T3 true T4 )
-* 语义分析 *-          完成
布尔表达式值 = true
```

5 出现的问题以及解决方案

5.1 词法分析

我对于词法分析的处理，没有使用课本中提供的 if...else if...else...结构，而是采用了 while 与 switch 结构，来达到在读取一个字符后实现状态转换的目的。

在实现过程中，我主要利用了 C++中的 vector（容器）数据结构作为字符缓冲区。当遇到终结态时，缓冲区中的字符为可识别字符，这时将其取出并以此为基础创建 Word 对象，将识别后的单词放入 wordsVector 中，然后清空缓冲区；而遇到中间态时，只需要将字符压入缓冲区并修改当前状态即可；在遇到非法字符时，主要分为两种情况，一种是在 0 状态下遇到非法租字符，则把以当前字符为基础创建 Word 对象，压入 wordsVector 中，另一种是在中间状态下遇到非法字符，则首先进行回溯，即只将缓冲区队头字符作为一个非法单词压入 wordsVector 中，其他字符归还到输入串中。

5.2 SLR 分析表的存储

对于 SLR 分析表的存储，我考虑过使用图，但最终决定了利用两张二维表来存储 SLR 分析表的全部信息。

其中，SLR 表存放下一步要转换的状态去向，正数表示目标状态，0 表示 acc，-1 表示无效即报错；SLR_S 表存放下一步要变换的动作，0 表示移进，-1 表示规约，1 表示 goto，acc 看做一种特殊的规约。

5.3 四元式的生成

我考虑到的问题是在表达式求值中，利用逆波兰式的计算过程，来模拟四元式的输出过程。这里遇到的主要问题是如何记录中间变量的序号，即 Tn。

由于我只使用了一个操作数栈来实现，所以在该算法中很难做到判断栈中出栈的操作数是中间变量 Tn 还是源操作数 id。因此我修改了 Word 的属性值，增加了 TempNumber 来表示信息，0 表示当前 Word 为源操作数，正数表示为第 n 个中间变量，从而解决了这个问题。

6 心得体会

6.1 缺陷

没有给出可视化的界面，并且词法分析中也没有很好的处理数字的问题，只是很简单的按照了正整数，负整数，正小数，负小数来划分。

6.2 展望

在之后的完善中，增加可视化界面，尝试引入多线程，达到更好的批处理效果，同时引入界面后，对于语法分析，将展示更加详细的过程，显示 Action 与 Goto。

6.3 感受

通过本次课程设计，我深刻理解了编译器从词法分析，到语法分析再到语义分析产生中间代码的整个工作流程。提高了我对此类问题的分析与设计能力，更加熟练地掌握了栈和队列等常用数据结构的运用，提高了我的编程能力。我在课程设计中也复习了逆波兰式的构造算法，表达式求值算法，参照 SLR 分析表进行语法分析的过程等常用算法。

但同时也暴露了我的问题分析不是很全面的不足，在没有找到最优解的情况下去实现，这也让我在之后的代码优化中吃尽了苦头。

在之后我会针对自己的软点进行训练，同时我也会进一步学习课本之后的知识，查阅相关资料，进一步提高自己的能力，将这门计算机专业核心课学精学通。