

学生学号	0121210680206	课程成绩	
------	---------------	------	--

武汉理工大学

课程设计

课程名称 编译原理

开课学院 计算机科学与技术学院

指导教师 饶文碧

姓 名 廖 星

专业班级 软件工程 ZY1201 班

2015 年 1 月 12 日

目 录

课程设计任务书	2
1 课程设计目的	3
2 课程设计内容及步骤	3
3 翻译方法	3
3.1 词法分析	3
3.2 语法分析	4
3.3 中间代码生成	5
4 文法及属性文法描述	5
4.1 文法描述	5
4.2 属性文法描述	5
5 LR 分析法	6
5.1 LR 分析法概述	6
5.2 识别活前缀的 DFA	6
5.3 SLR(1)分析表	7
5.4 分析过程	7
6 课程设计结果	10
7 程序评价及心得体会	11
参考文献	12
附录 程序源代码	13
课程设计成绩评定表	22

课程设计任务书

学生姓名： 廖 星

专业班级： 软件工程 ZY1201 班

指导教师： 饶文碧

工作单位： 计算机科学与技术学院

题目：简单计算器的设计（LR 分析法）

目的

通过设计、编制、调试一个简单计算器程序，加深对语法及语义分析原理的理解，并实现词法分析程序对单词序列的词法检查和分析。

设计内容及要求

算术表达式的文法：

〈无符号整数〉 ::= 〈数字〉 { 〈数字〉 }

〈标志符〉 ::= 〈字母〉 { 〈字母〉 | 〈数字〉 }

〈表达式〉 ::= [+ | -] 〈项〉 { 〈加法运算符〉 〈项〉 }

〈项〉 ::= 〈因子〉 { 〈乘法运算符〉 〈因子〉 }

〈因子〉 ::= 〈标志符〉 | 〈无符号整数〉 | ‘(’ 〈表达式〉 ‘)’

〈加法运算符〉 ::= + | -

〈乘法运算符〉 ::= * | /

1. 选择 LR 分析法完成以上任务，计算出表达式的结果。
2. 写出符合分析方法要求的文法，给出分析方法的思想，完成分析程序设计。
3. 编制好分析程序后，设计若干用例，上机测试并通过所设计的分析程序。

时间安排

第 19 周周一至周五（1 月 12 日～16 日）：分班实验，调试程序

指导教师签名：

2015 年 月 日

简单计算器的设计（LR 分析法）

1 课程设计目的

通过设计、编制、调试一个简单计算器程序，加深对语法及语义分析原理的理解，并实现词法分析程序对单词序列的词法检查和分析。

2 课程设计及步骤

本次课程设计需要使用 LR 分析法完成简单计算器的设计，其中算术表达式的文法如下：

〈无符号整数〉 ::= 〈数字〉 { 〈数字〉 }

〈标志符〉 ::= 〈字母〉 { 〈字母〉 | 〈数字〉 }

〈表达式〉 ::= [+ | -] 〈项〉 { 〈加法运算符〉 〈项〉 }

〈项〉 ::= 〈因子〉 { 〈乘法运算符〉 〈因子〉 }

〈因子〉 ::= 〈标志符〉 | 〈无符号整数〉 | ‘(’ 〈表达式〉 ‘)’

〈加法运算符〉 ::= + | -

〈乘法运算符〉 ::= * | /

本次课程设计分为如下步骤完成：

1. 根据题目要求的文法写出产生式；
2. 进行文法拓广，根据产生式画出识别活前缀的 DFA；
3. 根据 DFA 写出 LR(0)或 SLR(1)分析表；
4. 编写程序，对输入串进行分析；
5. 设计若干用例，上机测试并通过所设计的分析程序。

3 翻译方法

3.1 词法分析

词法分析是计算机科学中将字符序列转换为单词（Token）序列的过程。进行语法分析的程序或者函数叫作词法分析器（Lexical analyzer, 简称 Lexer），也叫扫描器（Scanner）。

词法分析器一般以函数的形式存在，供语法分析器调用。词法分析是编译过程中的第一个阶段，在语法分析前进行。也可以和语法分析结合在一起作为一遍，由语法分析程序调用词法分析程序来获得当前单词供语法分析使用。简化设计、改进编译效率、增加编译系统的可移植性。词法分析是编制一个读单词的过程，从输入的源程序中，识别出各个具有独立意义的单词，即基本保留字、标识符、常数、运算符、分隔符五大类。并依次输出各个单词的内部编码及单词符号自身值。单词的分类主要分为五类：

1. 关键字：由程序语言定义的具有固定意义的标识符，也称为保留字或基本字。
2. 标识符：用来表示程序中各种名字的字符串。
3. 常数：常数的类型一般有整型、实型、布尔型、文字型。
4. 运算符：如+、-、*、/等。
5. 界限符：如逗号、分号、括号等。

这里将词法分析程序设计成一个子程序，每当语法分析程序需要一个单词时，则调用该子程序。词法分析程序每调用一次，便从源程序文件中读入一些字符，直到识别出一个单词。

3.2 语法分析

语法分析是编译程序的核心部分，其主要任务是确定语法结构，检查语法错误，报告错误的性质和位置，并进行适当的纠错工作。

语法分析的主要工作是识别由词法分析给出的单词序列是否是给定的正确句子（程序）。语法分析常用的方法自顶向下的语法分析和自底向上的语法分析两大类。此次设计中语法分析中主要通过 LR 分析法对语法分析处理过程进行控制，使表达式结果能正确得出，同时识别语法分析中的语法错误。

语法分析是编译过程的一个逻辑阶段。语法分析的任务是在的基础上将单词序列组合成各类语法短语，如“程序”、“语句”、“表达式”等等。语法分析程序判断源程序在结构上是否正确。源程序的结构由上下文无关文法描述。语法分析程序可以用 YACC 等工具自动生成。

在语法分析的同时可由语法分析程序调用相应的语义子程序进行语义处理，完成附加在所使用的产生式上的语义规则描述，并完成移进—归约过程。

词法分析程序和语法分析程序的关系如图 1 所示：

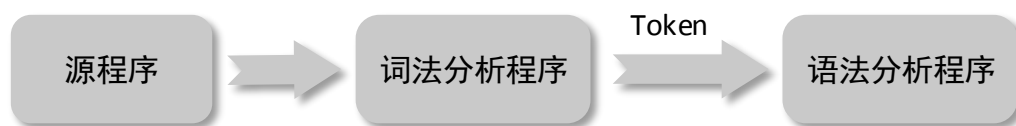


图 1 词法与语法分析的关系

3.3 中间代码生成

中间代码，也称中间语言，是复杂性介于源程序语言和机器语言的一种表示形式。为了使编译程序有较高的目标程序质量，或要求从编译程序逻辑结构上把与机器无关和与机器有关的工作明显的分开来时，许多编译程序都采用了某种复杂性介于源程序语言和机器语言之间的中间语言。中间代码（语言）是一种特殊结构的语言，编译程序所使用的中间代码有多种形式。按其结构分常见的有逆波兰式（后缀式）、三地址代码（三元式、四元式）和树形表示（抽象语法树）、DAG 表示。

本次课程设计不需要生成中间代码。

4 文法及属性文法描述

4.1 文法描述

根据题目要求的文法，产生式如下：

- (1) $S' \rightarrow S$ (文法拓广)
- (2) $S \rightarrow E$
- (3) $E \rightarrow E + T$
- (4) $E \rightarrow E - T$
- (5) $E \rightarrow T$
- (6) $T \rightarrow T * F$
- (7) $T \rightarrow T / F$
- (8) $T \rightarrow F$
- (9) $F \rightarrow (E)$
- (10) $F \rightarrow +i$
- (11) $F \rightarrow -i$
- (12) $F \rightarrow i$

4.2 属性文法描述

对该文法的属性文法描述如下：

产生式	语义规则
$S \rightarrow E$	$\text{print}(E.\text{val})$
$E \rightarrow E + T$	$E.\text{val} := E.\text{val} + T.\text{val}$
$E \rightarrow E - T$	$E.\text{val} := E.\text{val} - T.\text{val}$
$E \rightarrow T$	$\text{print}(T.\text{val})$
$T \rightarrow T * F$	$T.\text{val} := T.\text{val} * F.\text{val}$
$T \rightarrow T / F$	$T.\text{val} := T.\text{val} / F.\text{val}$
$T \rightarrow F$	$\text{print}(F.\text{val})$
$F \rightarrow (E)$	$F.\text{val} := E.\text{val}$
$F \rightarrow +i$	$F.\text{val} := i.\text{lexval}$
$F \rightarrow -i$	$F.\text{val} := -i.\text{lexval}$
$F \rightarrow i$	$F.\text{val} := i.\text{lexval}$

5 LR 分析法

5.1 LR 分析法概述

LR 分析法是一种能根据当前分析栈中的符号串（通常以状态表示）和向右顺序查看输入串的 k 个（ $k \geq 0$ ）符号就可以唯一地确定分析器的动作是移进还是归约和用哪个产生式归约，因而也就能唯一地确定句柄。LR 分析法的归约过程是规范推导的逆过程，所以 LR 分析过程是一种规范归约的过程。

LR(k)分析方法是 1965 年 Knuth 提出的，括号中的 k 表示向右查看输入串符号的个数。这种方法比起自顶向下的 LL(k)分析方法和自底向上的优先分析方法对文法的限制要少得多，也就是说对于大多数用无二义性上下文无关文法描述的语言都可以用相应的 LR 分析器进行识别，而且这种方法还具有分析速度快，能准确、即时地指出出错位置。它的主要缺点是对于一个实用语言文法的分析器的构造工作量相当大， k 愈大构造愈复杂，实现比较困难。因此，目前许多实用的编译程序，都运用了 LR 分析器。

5.2 识别活前缀的 DFA

本课程设计根据产生式得到的识别活前缀的 DFA 如图 2 所示：

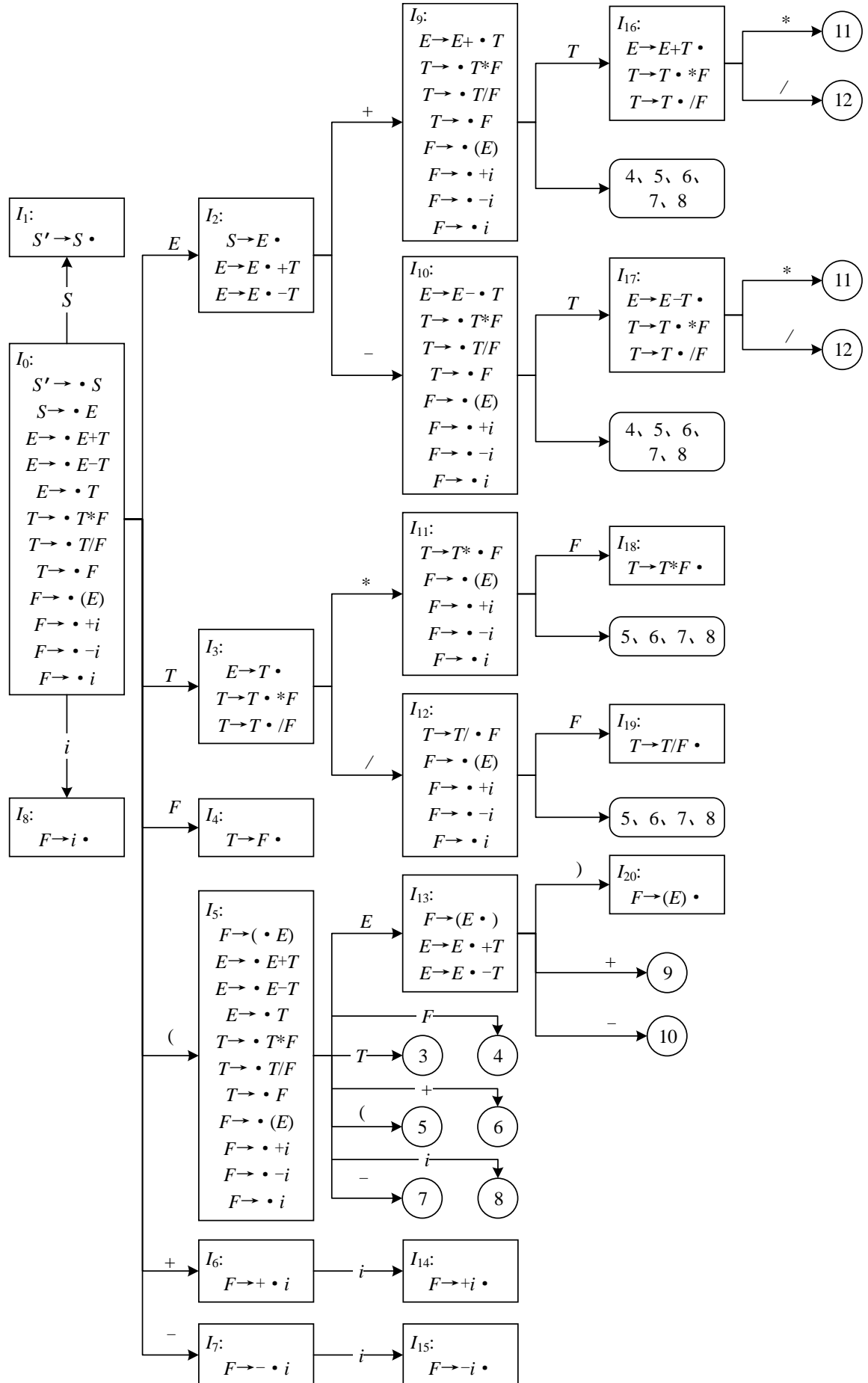


图2 识别活前缀的DFA

5.3 SLR(1)分析表

根据图 2 可知, I_2 、 I_3 、 I_{16} 、 I_{17} 存在移进—归约冲突, 可用 SLR(1)方法解决冲突。

SLR(1)分析表如下:

	ACTION								GOTO			
	+	-	*	/	()	i	#	<i>S</i>	<i>E</i>	<i>T</i>	<i>F</i>
0	S_6	S_7			S_5		S_8		1	2	3	4
1								<i>acc</i>				
2	S_9	S_{10}						r_2				
3	r_5	r_5	S_{11}	S_{12}		r_5		r_5				
4	r_8	r_8	r_8	r_8		r_8		r_8				
5	S_6	S_7			S_5		S_8			13	3	4
6							S_{14}					
7							S_{15}					
8	r_{12}	r_{12}	r_{12}	r_{12}		r_{12}		r_{12}				
9	S_6	S_7			S_5		S_8				16	4
10	S_6	S_7			S_5		S_8				17	4
11	S_6	S_7			S_5		S_8					18
12	S_6	S_7			S_5		S_8					19
13	S_9	S_{10}				S_{20}						
14	r_{10}	r_{10}	r_{10}	r_{10}		r_{10}		r_{10}				
15	r_{11}	r_{11}	r_{11}	r_{11}		r_{11}		r_{11}				
16	r_3	r_3	S_{11}	S_{12}		r_3		r_3				
17	r_4	r_4	S_{11}	S_{12}		r_4		r_4				
18	r_6	r_6	r_6	r_6		r_6		r_6				
19	r_7	r_7	r_7	r_7		r_7		r_7				
20	r_9	r_9	r_9	r_9		r_9		r_9				

5.4 分析过程

LR 分析法的规约过程是规范推到的逆过程, 所以 LR 分析过程是一种规范规约的过

程。其分析过程为：由文法构造出该文法项目集，再根据项目集构造该文法的 DFA，再判断是否有移进—规约和规约—规约冲突，若没有冲突则该文法为 LR(0)的，若有冲突则该文法是 SLR(1)的，最后可以构造出 LR(0)分析表。然后根据 LR(0)分析表进行语法分析，分析过程就是进栈和规约的过程。若能规约出开始符 S ，则语法正确。反之，语法错误。

LR 分析法过程流程图如下：

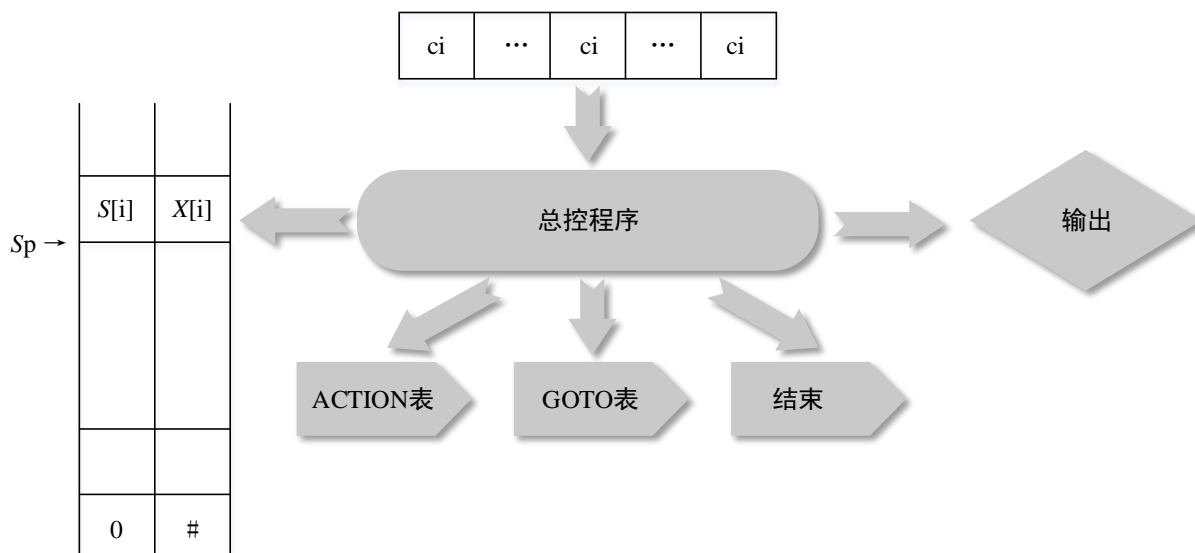


图 3 LR 分析法流程图

其中 Sp 为栈顶指针， $S[i]$ 为状态栈， $X[i]$ 为文法符号栈。状态转换表内容按关系 $GOTO[S_i, X] = S_j$ 确定，改关系式是指当前栈顶状态为 S_i 遇到当前文法符号为 X 时应转向状态 S_j 。 X 为终结符或非终结符。

$ACTION[S_i, a]$ 规定了栈顶状态为 S_j 时遇到输入符号 $c[i]$ 应该执行的动作。动作有以下四种可能：

移进：当 $S_j = GOTO[S_i, a]$ 成立，则把 S_j 移入到文法符号栈。其中 i, j 表示状态号。

规约：当在栈顶形成句柄为 b 时，则用 b 归约为相应的非终结符 A ，即当文法中有 $A \rightarrow b$ 的产生式，而 b 的长度为 r ，则从状态栈和文法符号栈中自栈顶向下去掉 r 个符号。并把 A 移入文法符号栈内，再把满足 $S_j = GOTO[S_i, A]$ 的状态移进状态栈，其中 S_i 为修改指针后的栈顶状态。

接受：当归约到文法符号栈中只剩下文法的开始符号 S 时，并且输入符号串已

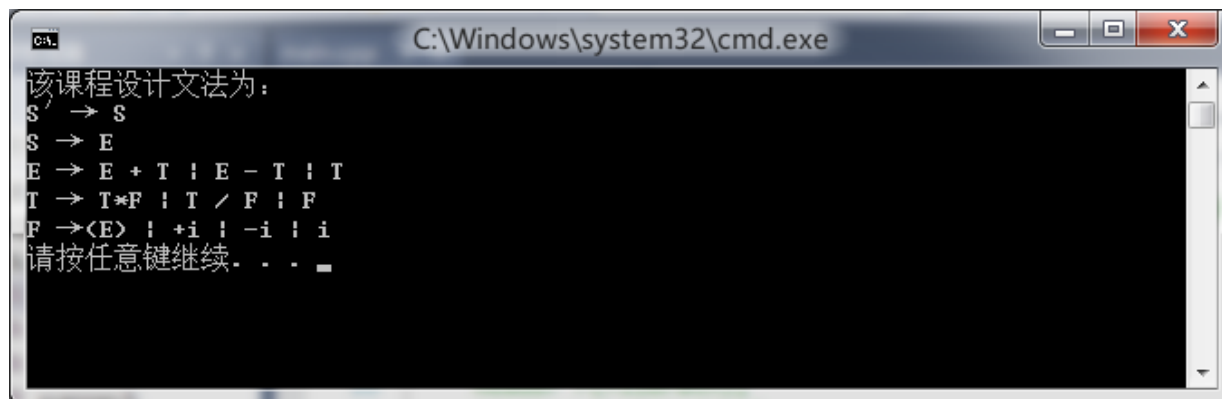
结束：即当前输入符是 ‘#’，则为分析成功。

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

入串不是该分发能接受的句子。

6 课程设计结果

本次课程设计采用的文法和产生式：



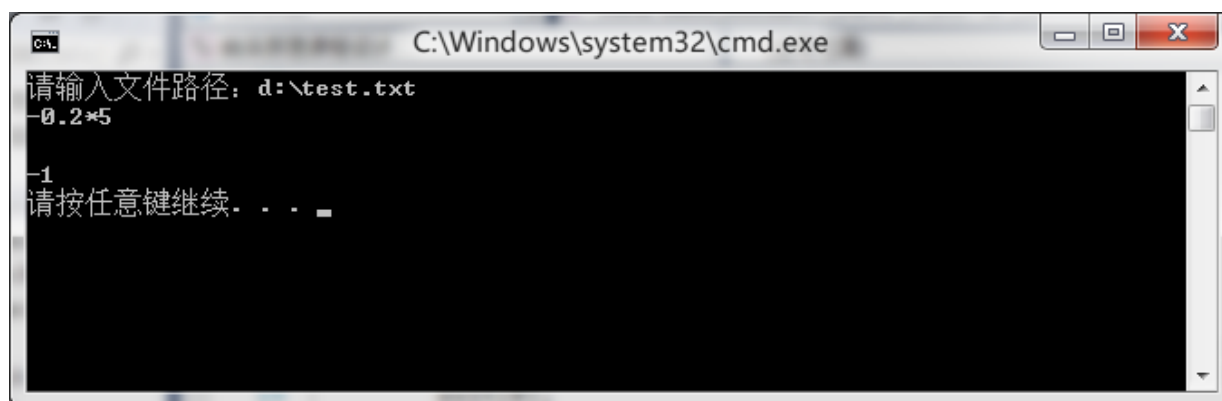
```

C:\Windows\system32\cmd.exe
该课程设计文法为：
S → S
S → E
E → E + T | E - T | T
T → T * F | T / F | F
F → (E) | +i | -i | i
请按任意键继续. . .
    
```

图 4 文法和产生式

本次课程设计取了多组测试用例，测试结果如下：

-0.2*5

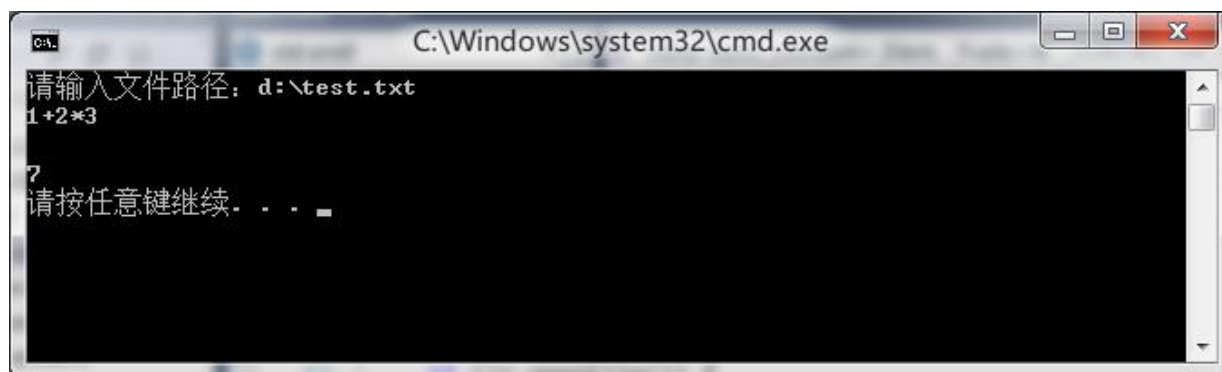


```

C:\Windows\system32\cmd.exe
请输入文件路径: d:\test.txt
-0.2*5
-1
请按任意键继续. . .
    
```

图 5 运行结果 1

1+2*3



```

C:\Windows\system32\cmd.exe
请输入文件路径: d:\test.txt
1+2*3
7
请按任意键继续. . .
    
```

图 6 运行结果 2

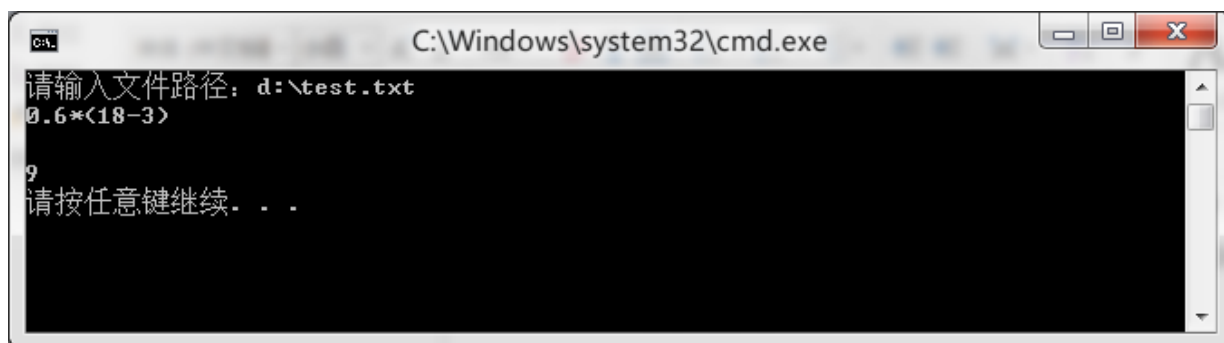
$0.6*(18-3)$ 

图 7 运行结果 3

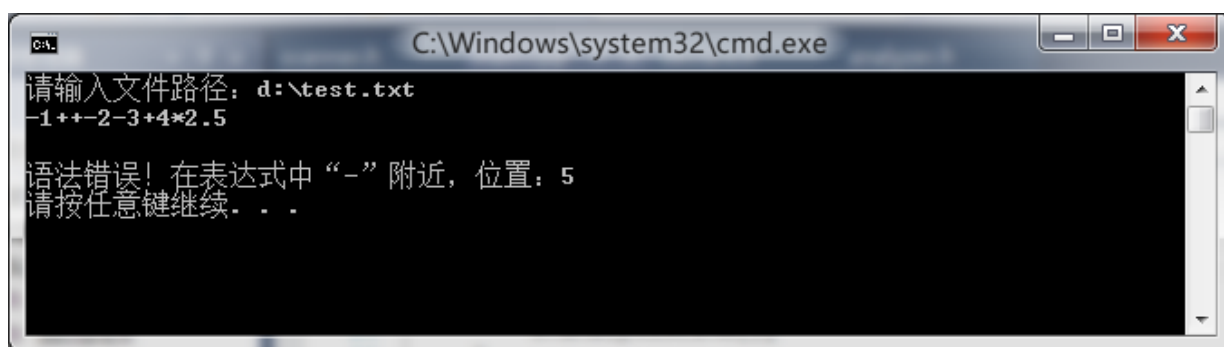
 $-1++-2-3+4*2.5$ 

图 8 运行结果 4

前三个测试用例输入的是正确的表达式，程序的得出的计算结果也正确。最后一个是错误的表达式，第 4 个位置的加号根据产生式 10 被识别为正号，而后面的减号（或负号）无法被识别，因而程序报错。

7 程序评价及心得体会

本次课程设计使用 LR 分析法完成简单计算器的实现。通过对文法的分析，写出了相应产生式，并完成了识别活前缀的 DFA 的绘制、SLR(1)分析表的填写，以及编程实现对输入串的分析过程。并且对题目要求进行了扩展，在词法分析环节加入了对小数的识别，使该计算器支持小数运算。最终通过测试，可验证产生式、分析表和程序的正确性。

本次课程设计通过文件读取将文件内容读入到内存中。由于考虑到可能存在的格式问题，词法分析环节会过滤掉首部和尾部空格，再对输入串进行分析。

通过本次课程设计，我对 LR 分析法有了更深刻的认识，感到受益匪浅。

参考文献

- [1] 张素琴, 吕映之, 蒋维杜, 戴桂兰. 编译原理 (第二版) [M]. 北京: 清华大学出版社 2005.2.
- [2] [美] Alfred V.Aho, Monica S.Lam, Ravi Sethi, Jeffrey D.Ullman. 编译原理 [M]. 北京: 机械工业出版社, 2009.1.1.
- [3] [美] Andrew W.Appel. 现代编译原理: C 语言描述 [M]. 北京: 人民邮电出版社, 2006.4.1.
- [4] [美] Steven S.Muchnick 著, 赵克佳, 沈志宇译. 高级编译器设计与实现 [M]. 北京: 机械工业出版社, 2005.7.3.

附录 程序源代码

analyzer.h

```

class Analyzer {
private:
    vector<Word>::size_type
index;
    stack<int> state;
    stack<Word> symbol;
    Word& readWord();
    int readState();
    SLR& compare(int state,
Word& input);
    void shift(int state, Word&
input);
    void reduce(int index);
    double calculate(double num1,
const string& operate, double
num2);
    void error(const string& msg,
vector<Word>::size_type index =
0);
    void pop();
public:
    Analyzer();
    void show();
    void analySyntax();
};

Analyzer::Analyzer() : state(),
symbol() {
    index = 0;
}

Word& Analyzer::readWord() {
    return words[index];
}

}

int Analyzer::readState() {
    return state.top();
}

void Analyzer::analySyntax() {
    state.push(0);
    symbol.push(Word(SRP, "#"));
    while (true) {
        int cur_state =
readState();
        Word cur_input =
readWord();
        SLR result =
compare(cur_state, cur_input);
        switch (result.getType())
        {
            case 0:
                error("语法错误! 在表达
式中" + cur_input.getWord() + "
附近, 位置: ", index);
                break;
            case 1:
                shift(result.getValue(),
cur_input);
                break;
            case 2:
                reduce(result.getValue());
            case 3:
                state.push(compare(state.top(),
symbol.top()).getValue());
                break;
            case 4:

```

```

cout << symbol.top().getValue()
<< endl;

        return;
    default:
        break;
    }
}

SLR& Analyzer::compare(int
state, Word& input) {
    return slr_table[state]
[input.getType()];
}

void Analyzer::shift(int state,
Word& input) {
    this->state.push(state);
    this->symbol.push(input);
    ++index;
}

void Analyzer::reduce(int index)
{
    switch (index) {
    case 2:
    case 5:
    case 8:
    case 12: {
        double num =
symbol.top().getValue();
        this->pop();
symbol.push(Word(left_part[ind
ex], num));
    }

```

```

        break;
    case 10:
    case 11: {
        double num =
symbol.top().getValue();
        this->pop();
        string s =
symbol.top().getWord();
        if (s == "-") num = -num;
        this->pop();
symbol.push(Word(left_part[ind
ex], num));
    }
        break;
    case 9: {
        this->pop();
        double num =
symbol.top().getValue();
        this->pop();
        this->pop();
symbol.push(Word(left_part[ind
ex], num));
    }
        break;
    case 3:
    case 4:
    case 6:
    case 7: {
        double num1 =
symbol.top().getValue();
        this->pop();
        string s =
symbol.top().getWord();
        this->pop();
        double num2 =

```

```

symbol.top().getValue();
    this->pop();
symbol.push(Word(left_part[index], calculate(num2, s, num1)));
    }
    break;
default:
    break;
    }
}

```

double

```

Analyzer::calculate(double
num1, const string& operate,
double num2) {
    switch (operate[0]) {
    case '+':
        return num1 + num2;
    case '-':
        return num1 - num2;
    case '*':
        return num1 * num2;
    case '/':
        if (num2 == 0) error("除
数不能为 0!");
        return num1 / num2;
    default:
        return 0;
    }
}

```

```

void Analyzer::error(const
string& msg,
vector<Word>::size_type index)
{

```

```

    if (index == 0) cout << msg <<
endl;
    else cout << msg << (index +
1) << endl;
    exit(0);
}

```

```

void Analyzer::pop() {
    symbol.pop();
    state.pop();
}

```

```

void Analyzer::show() {
    cout << "该课程设计文法为: " <<
endl;
    cout << "S'→ S" << endl;
    cout << "S → E" << endl;
    cout << "E → E + T | E - T |
T" << endl;
    cout << "T → T*F | T / F | F"
<< endl;
    cout << "F →(E) | +i | -i | i"
<< endl;
}

```

declare.h

```

enum Identifier {
    ADD,          // 加号、正号 +
    SUB,          // 减号、负号 -
    MUL,          // 乘号 *
    DIV,          // 除号 /
    LPR,          // 左括号 (
    RPR,          // 右括号 )
    INT,          // 整数
    SRP,          // 结束符 #

```

S,	SLR(0, 0), SLR(0, 0), SLR(0, 0),
E,	SLR(1,14), SLR(0, 0), SLR(0, 0),
T,	SLR(0, 0), SLR(0, 0), SLR(0, 0),
F	SLR(0, 0), SLR(0, 0), SLR(0, 0),
};	SLR(0, 0), SLR(0, 0), SLR(0, 0),
	SLR(1,15), SLR(0, 0), SLR(0, 0),
int left_part[] = { 0, 0, S, E,	SLR(0, 0), SLR(0, 0), SLR(0, 0),
E, E, T, T, T, F, F, F, F };	SLR(2,12), SLR(2,12), SLR(2,12),
vector<Word> words;	SLR(2,12), SLR(0, 0), SLR(2,12),
SLR slr_table[21][12] = {	SLR(0, 0), SLR(2,12), SLR(0, 0),
SLR(1, 6), SLR(1, 7), SLR(0, 0),	SLR(0, 0), SLR(0, 0), SLR(0, 0),
SLR(0, 0), SLR(1, 5), SLR(0, 0),	SLR(1, 6), SLR(1, 7), SLR(0, 0),
SLR(1, 8), SLR(0, 0), SLR(3, 1),	SLR(0, 0), SLR(1, 5), SLR(0, 0),
SLR(3, 2), SLR(3, 3), SLR(3, 4),	SLR(1, 8), SLR(2, 2), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),	SLR(0, 0), SLR(3,16), SLR(3, 4),
SLR(0, 0), SLR(0, 0), SLR(0, 0),	SLR(1, 6), SLR(1, 7), SLR(0, 0),
SLR(0, 0), SLR(4, 0), SLR(0, 0),	SLR(0, 0), SLR(1, 5), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),	SLR(1, 8), SLR(2, 2), SLR(0, 0),
SLR(1, 9), SLR(1,10), SLR(0, 0),	SLR(0, 0), SLR(3,17), SLR(3, 4),
SLR(0, 0), SLR(0, 0), SLR(0, 0),	SLR(1, 6), SLR(1, 7), SLR(0, 0),
SLR(0, 0), SLR(2, 2), SLR(0, 0),	SLR(0, 0), SLR(1, 5), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),	SLR(1, 8), SLR(2, 2), SLR(0, 0),
SLR(2, 5), SLR(2, 5), SLR(1,11),	SLR(0, 0), SLR(0, 0), SLR(3,18),
SLR(1,12), SLR(0, 0), SLR(2, 5),	SLR(1, 6), SLR(1, 7), SLR(0, 0),
SLR(0, 0), SLR(2, 5), SLR(0, 0),	SLR(0, 0), SLR(1, 5), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),	SLR(1, 8), SLR(2, 2), SLR(0, 0),
SLR(2, 8), SLR(2, 8), SLR(2, 8),	SLR(0, 0), SLR(0, 0), SLR(3,19),
SLR(2, 8), SLR(0, 0), SLR(2, 8),	SLR(1, 9), SLR(1,10), SLR(0, 0),
SLR(0, 0), SLR(2, 8), SLR(0, 0),	SLR(0, 0), SLR(0, 0), SLR(1,20),
SLR(0, 0), SLR(0, 0), SLR(0, 0),	SLR(0, 0), SLR(0, 0), SLR(0, 0),
SLR(1, 6), SLR(1, 7), SLR(0, 0),	SLR(0, 0), SLR(0, 0), SLR(0, 0),
SLR(0, 0), SLR(1, 5), SLR(0, 0),	SLR(2,10), SLR(2,10), SLR(2,10),
SLR(1, 8), SLR(0, 0), SLR(0, 0),	SLR(2,10), SLR(0, 0), SLR(2,10),
SLR(3,13), SLR(3, 3), SLR(3, 4),	SLR(0, 0), SLR(2,10), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),	SLR(0, 0), SLR(0, 0), SLR(0, 0),

```
SLR(2,11), SLR(2,11), SLR(2,11),
SLR(2,11), SLR(0, 0), SLR(2,11),
SLR(0, 0), SLR(2,11), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),
SLR(2, 3), SLR(2, 3), SLR(1,11),
SLR(1,12), SLR(0, 0), SLR(2, 3),
SLR(0, 0), SLR(2, 3), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),
SLR(2, 4), SLR(2, 4), SLR(1,11),
SLR(1,12), SLR(0, 0), SLR(2, 4),
SLR(0, 0), SLR(2, 4), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),
SLR(2, 6), SLR(2, 6), SLR(2, 6),
SLR(2, 6), SLR(0, 0), SLR(2, 6),
SLR(0, 0), SLR(2, 6), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),
SLR(2, 7), SLR(2, 7), SLR(2, 7),
SLR(2, 7), SLR(0, 0), SLR(2, 7),
SLR(0, 0), SLR(2, 7), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0),
SLR(2, 9), SLR(2, 9), SLR(2, 9),
SLR(2, 9), SLR(0, 0), SLR(2, 9),
SLR(0, 0), SLR(2, 9), SLR(0, 0),
SLR(0, 0), SLR(0, 0), SLR(0, 0)
};
```

reader.h

```
class Reader {
private:
    string filePath;
    ifstream ifs;
public:
    Reader();
    Reader(const string
filePath);
```

```
bool openFile();
bool hasNext();
string getLine();
~Reader();
};

Reader::Reader() {}

Reader::Reader(const string
filePath) {
    this->filePath = filePath;
}

bool Reader::openFile() {
    if (!ifs.is_open())
ifs.open(filePath.c_str(),
ios::in);
    return ifs.good();
}

bool Reader::hasNext() {
    return !ifs.fail();
}

string Reader::getLine() {
    string str;
    getline(ifs, str);
    return str;
}

Reader::~~Reader() {
    ifs.close();
}
```

scanner.h

```

class Scanner {
private:
    int row;
    string token;
    Reader& reader;
public:
    Scanner(Reader& reader);
    bool hasNextLine();
    void scanLine(const string&
str, string::size_type& index);
    void analysisLine();
    void out(int type, const
string& word);
};

Scanner::Scanner(Reader&
reader) : reader(reader) {
    row = 1;
}

bool Scanner::hasNextLine() {
    return reader.hasNext();
}

void Scanner::scanLine(const
string& str, string::size_type&
index) {
    if (index >= str.size())
return;
    char ch = str[index++];
    if (ch == '.') {
        cout << "小数点前必须是数字,
位置: " << index << endl;
        exit(0);
    }

```

```

    else if (isdigit(ch)) {
        token = "";
        while (isdigit(ch) || ch
== '.') {
            token.push_back(ch);
            ch = str[index++];
        }
        --index;
        out(INT, token);
    }
    else {
        switch (ch) {
            case '+':
                out(ADD, "+");
                break;
            case '-':
                out(SUB, "-");
                break;
            case '*':
                out(MUL, "*");
                break;
            case '/':
                out(DIV, "/");
                break;
            case '(':
                out(LPR, "(");
                break;
            case ')':
                out(RPR, ")");
                break;
            case '#':
                out(SRP, "#");
                break;
            default:
                cout << "非法字符: " <<

```

<pre> ch << "", 位置: " << index << endl; exit(0); break; } } void Scanner::analysisLine() { string line = reader.getLine(); string::size_type index = 0; cout << line << endl; while (index != line.size()) { while (ispace(line[index++])); this->scanLine(line, --index); } ++this->row; } void Scanner::out(int type, const string& word) { /* 记录词法分析结果 */ words.push_back(Word(type, word)); /* 输出词法分析结果 */ //cout << "(" << type << ", " << word << ")" << endl; }; slr.h class SLR { </pre>	<pre> private: int type; int value; public: SLR(); SLR(int type, int value); int getType(); void setType(int type); int getValue(); void setValue(int value); }; SLR::SLR() {} SLR::SLR(int type, int value) { setType(type); setValue(value); } int SLR::getType() { return type; } void SLR::setType(int type) { this->type = type; } int SLR::getValue() { return this->value; } void SLR::setValue(int value) { this->value = value; } </pre>
--	--

stdafx.h

```
#include <iostream>
#include <fstream>
#include <string>
#include <cctype>
#include <cstdlib>
#include <vector>
#include <stack>
```

word.h

```
class Word {
private:
    int type;
    string word;
    double value;
public:
    Word();
    Word(int type, string word);
    Word(int type, double
value);
    int getType();
    void setType(int type);
    string getWord();
    void setWord(string word);
    double getValue();
    void setValue(double value);
};

Word::Word() {}

Word::Word(int type, string word)
{
    setType(type);
    setWord(word);
}
```

```
Word::Word(int type, double
value) {
    setType(type);
    setValue(value);
}

int Word::getType() {
    return type;
}

void Word::setType(int type) {
    this->type = type;
}

string Word::getWord() {
    return word;
}

void Word::setWord(string word)
{
    this->word = word;
    if (isdigit(word[0]))
this->value =
atof(word.c_str());
}

double Word::getValue() {
    return this->value;
}

void Word::setValue(double
value) {
    this->value = value;
}
```


main.cpp

```
#include "stdafx.h"
#include "word.h"
#include "slr.h"
#include "declare.h"
#include "reader.h"
#include "scanner.h"
#include "analyzer.h"

using namespace std;

int main() {
    string filePath;
    cout << "请输入文件路径: ";
    getline(cin, filePath);
    Reader r(filePath);
```

```
    if (!r.openFile()) {
        cout << "未找到此文件" <<
endl;
        exit(0);
    }
    Scanner s(r);
    while (s.hasNextLine()) {
        s.analysisLine();
    }
    s.out(SRP, "#");
    Analyzer a;
    a.show();
    a.analySyntax();
    return 0;
}
```


课程设计成绩评定表

班级：软件工程 ZY1201 班

姓名：廖星

学号：0121210680206

序号	评分项目	满分	实得分
1	学习态度认真、遵守纪律	10	
2	设计分析合理性	10	
3	设计方案正确性、可行性	20	
4	设计结果正确性	40	
5	设计报告的规范性	10	
6	设计验收	10	
		总得分 / 等级	
评语：			

注：最终成绩以五级分制记。优（90—100 分）、良（80—89 分）、中（70—79 分）、及格（60—69 分）、60 分以下为不及格

指导教师签名：

年 月 日

