

北京交通大学
BEIJING JIAOTONG UNIVERSITY

《编译原理》 实验报告

实验名称:	专题 3_LL(1)语法分析设计原理与实现
学 号:	
姓 名:	
学 院:	计算机与信息技术学院
日 期:	2022 年 10 月 20 日

目录

1.	实验目的	3
2.	实验要求	3
3.	程序实现	3
3.1.	相关环境介绍	3
3.2.	主要数据结构	3
3.3.	程序结构描述	4
3.3.1.	设计方法	4
3.3.2.	函数定义	4
4.	程序测试	5
5.	实验汇总	7
5.1.	技术难点及解决方案	7
5.2.	实验感想和经验总结	8

1. 实验目的

通过实验，掌握 LL（1）文法及其判定；掌握 LL(1)分析表构造、分析和分析器的构造。

2. 实验要求

完成四则运算描述赋值语句的 LL(1)文法分析，实现 LL(1)分析中控制程序（表驱动程序）；完成以下描述赋值语句文法的 LL(1)分析过程。（1）输入串应是词法分析的输出二元式序列，即某算术表达式“专题 1”的输出结果。输出为输入串是否为该文法定义的算术表达式的判断结果；（2）LL(1)分析过程应能发现输入串出错；（3）设计两个测试用例（尽可能完备，正确和出错），并给出测试结果；（4）考虑根据 LL(1)文法编写程序构造 LL（1）分析表，包括 FIRST 集合和 FOLLOW 集合，并添加到你的 LL（1）分析程序中。

3. 程序实现

3.1. 相关环境介绍

操作系统：window 10 21H2

开发环境：Clion-2022.2.1-Windows

编译器：mwing-10.0

3.2. 主要数据结构

主要是单词的信息保存，建立了一个 struct

```
01: struct Keyword{
02:     string notation;
03:     int class_num;
04:     int line;
05:     Keyword(string str, int num, int line_){
06:         notation = str;
07:         class_num = num;
08:         line = line_;
09:     }
10:     Keyword(char* str, int num, int line_){
11:         notation = string(str);
12:         class_num = num;
13:         line = line_;
```

```

14:     }
15:     keyword(char str, int num, int line_){
16:         notation = str;
17:         class_num = num;
18:         line = line_;
19:     }
20:};

```

其中 notation 为单词的值，class_num 为单词所属的类别，line 是单词在源程序中的行号。利用 map 容器，制作了以 string 为 index 的二维数组。

```

21:map<string, map<string, vector<string>>> analyzer_table;
22:map<string, set<string> > first;
23:map<string, set<string> > follow;
24:map<string, string> Vn;
25:map<string, string> Vt;
26:map<string, int> flag_e;
27:map<string, vector<string>[ORMAXFORGRAMMAR] > grammar;
28:map<pair<string,vector<string>>, set<string>> first_alpha;

```

3.3. 程序结构描述

3.3.1. 设计方法

考虑到程序的通用性，程序的输入流全部都是 txt 文件。首先读入 grammar 文件，其中包括文法和 Vn 以及 Vt 集的定义。之后，在程序中创建 Vn 集合 Vt 集，然后求的 first 集合 follow 集，在这基础上构建出 LL(1)分析表。

接下来是和词法分析程序的联动，本程序可以直接调用词法分析程序，因此测试样例直接准备源代码待分析即可。当然，语法分析部分当然还是词法分析的结果。其中的相应 config 配置依赖文件参考实验 1。

3.3.2. 函数定义

create_Vn_Vt_grammar(grammar_path); 创建 Vn Vt 集和文法表。
 create_first(grammar_path); 创建 first 集。
 create_follow(); 创建 follow 集
 create_analyzer_table(); 创建分析表
 analyzer_stack(need_to_analysis); 分析栈入口，即分析驱动程序。

int correct_prom() 语句符合文法提示。

int error_prom() 语句不合文法提示。

还有一众辅助函数，不表。

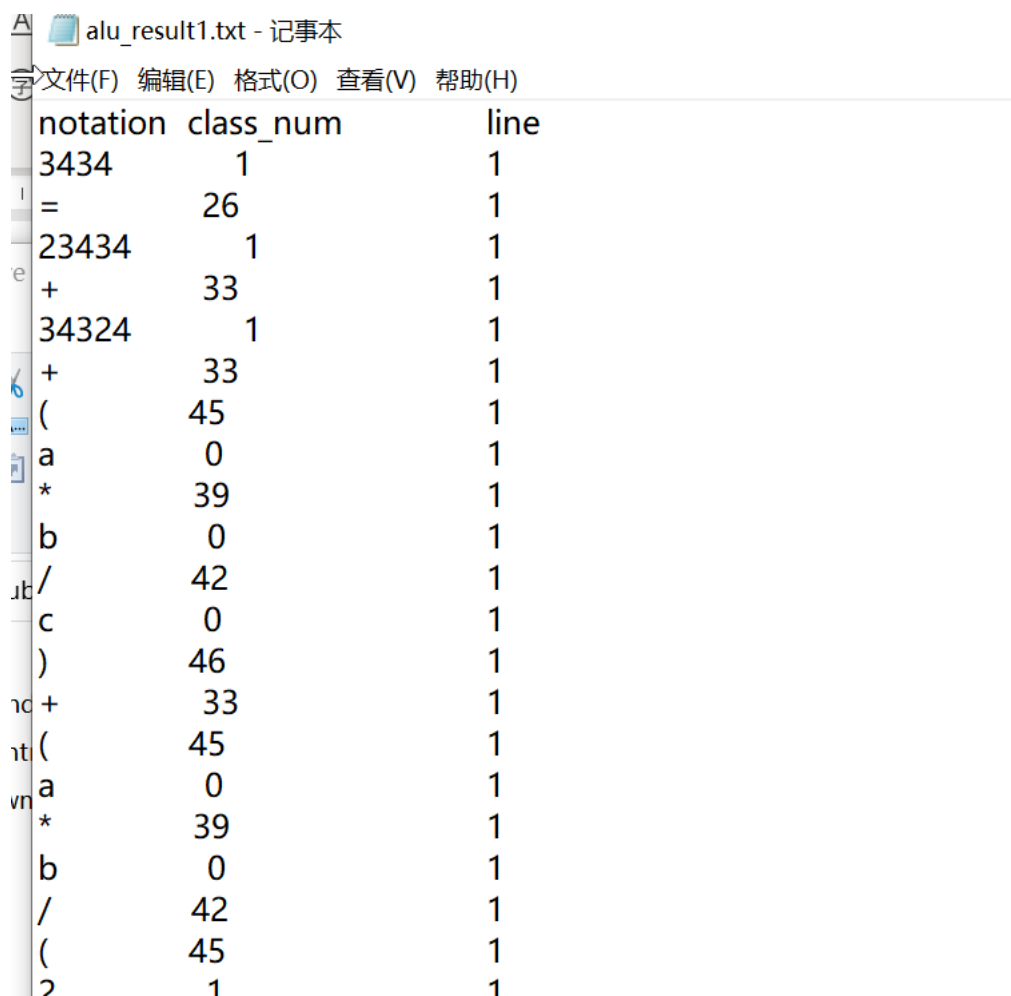
主要函数的编写参照课程讲述。

4. 程序测试

Alu_test1.txt

这里一个文件中包含了 4 个测试样例。

```
01:3434 = 23434+34324+(a*b / c) +(a*b/ (2-2)) #
02:a = ( a+ b -c ) * 32 / a #
03:b=(a+c * b #
04:k=a- b *c #
```



notation	class_num	line
3434	1	1
=	26	1
23434	1	1
+	33	1
34324	1	1
+	33	1
(45	1
a	0	1
*	39	1
b	0	1
/	42	1
c	0	1
)	46	1
+	33	1
(45	1
a	0	1
*	39	1
b	0	1
/	42	1
(45	1
2	1	1

图表 4-1 test1 中词法分析结果

alu_class_base_result.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

notation	class_num	line
+	33	1
-	36	1
*	39	1
/	42	1
(45	1
)	46	1
#	24	1
=	26	1

图表 4-2 符号类号文件（词法分析程序的输出结果的一部分）

```
base_source ../alu_class_base_test.txt
base_out ../alu_class_base_result.txt
input_file ../alu_test1.txt
output_file ../alu_result1.txt

// PATH have something problems
// path have blank so fin stop after junior
// path recommend to use compared path case the absolute path has blank
// base on reversed_word.txt
```

图表 4-3config 文件

其中包括了一众依赖文件的路径还有默认待分析文件路径，最下边是书写格式。

```

1  Vn S E E' T T' F A M V
2  Vt + - * / = ( ) i n
3
4  S -> V = E
5  E -> T E'
6  E' -> A T E' | e
7  T -> F T'
8  T' -> M F T' | e
9  F -> ( E ) | i | n
10 A -> + | -
11 M -> * | /
12 V -> i
13
14 end
15 // e is epilog
16 // each line ends with nothing, no blank
17 // Vt i is identifier num
18

```

图表 4-4 grammar 文件

Grammar 文件主要是 Vn Vt 集还有文法的输入，下方是特殊说明。

词法分析程序输出仍然采用了三元组，多了一个行号，方便错误提示，更多的我们直接用了 Keyword 结构，这对结果没有影响。

```

"D:\BJTU\subject\Junior First\Compile Princi
line 1 has syntax error
line 2 syntax correct
line 3 has syntax error
line 4 syntax correct
Process finished with exit code 0

```

图表 4-5 本程序结果，信息提示

通过信息提示可以看出，程序对 4 个测试样例完成的很好，结果正确。其中等号左侧只能是标识符，右侧实验指导书原文法标识符位置，可以出现数字常量。

5. 实验汇总

5.1. 技术难点及解决方案

实验本身的重要程序框图，已经在课堂上教授过了，主要是代码编写过程中的问题有些多，所幸最后都解决了。特别是 first 集和 follow 集的创建，人工建立和代码编写还是有一定的区别的，许多人工的取巧方法，机器都无法使用，这里也又一次感受到了程序编写

的形式化魅力。

这里的实验和课堂上讲述的就是多了一个赋值语句，但是给的也十分的简单，如果 $V \rightarrow E$ 这样就是一般的情况，实验要求的是 $V \rightarrow I$ 这样会简单一些，直接判断就可以了。

实验代码编写完成之后，本人突然发现，实验指导的文法与正常编程语言的语法逻辑还是有一定区别的。其中 i 被定义为标识符，也就是变量，这在赋值文法的等号左侧是成立的，但是等号右侧的 i 出现的位置可以是标识符也可以是数字常量，甚至可以是一个表达式。这里我们在原有文法的基础上做了一点拓展，新增了一个 n 的 Vt 符号，作为数字常量的代指。这样我们的文法就变成了：

$$\begin{aligned} S &\rightarrow V = E \\ E &\rightarrow T E' \\ E' &\rightarrow A T E' \mid e \\ T &\rightarrow F T' \\ T' &\rightarrow M F T' \mid e \\ F &\rightarrow (E) \mid i \mid n \\ A &\rightarrow + \mid - \\ M &\rightarrow * \mid / \\ V &\rightarrow i \end{aligned}$$

幸运的是，在修改文法后本人的实验 3 程序居然没有一点问题，这真是值得欣慰的事。

当然其实我们也可以把 i 再拓展为表达式，这样其实是一种递归调用。

5.2. 实验感想和经验总结

这次实验主要的编码困难在于数据结构的建立。C++ 中的二维数组 `index` 都是整数，课堂上教授的分析表等表格都是字符作为 `index`，这其中的转换是有难度的。最后，想到了使用 STL 中的 `map` 容器作为数据结构，二维数组就是 `map` 套 `map`，还用了 `vector` 来存储文法产生式，这样就会出现“**数据结构**”一节中看似很复杂的容器嵌套结构。数据结构敲定之后，再定义出基本的操作方法，算法沿用课程中教授的，实验就不难完成了。