

北京交通大学
BEIJING JIAOTONG UNIVERSITY

《编译原理》 实验报告

实验名称:	专题 4 算符优先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 11 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	5
4.	程序测试	5
5.	实验汇总	9
5.1.	技术难点及解决方案	9
5.2.	实验感想和经验总结	9

1. 实验目的

通过实验，掌握并实现算符优先分析算法，完成算术表达式的算符优先文法的算符优先分析过程。

2. 实验要求

- (1) 构造该算符优先文法的优先关系矩阵或优先函数；
- (2) 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。
- (3) 算符优先分析过程应能发现输入串出错。
- (4) 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；
- (5) 考虑根据算符优先文法构造算符优先关系矩阵，包括 FIRSTVT 和 LASTVT 集合，并添加到你的算符优先分析程序中。

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。

下边是以 stl 为基础建立的相关数据结构

```

21:map<string, map<string, int>> analyzer_table;
22:map<string, set<string> > firstVt;
23:map<string, set<string> > lastVt;
24:map<string, string> Vn;
25:map<string, string> Vt;
26:
27:map<string, vector<string>[ORMAXFORGRAMMAR] > grammar;
28:map<pair<string,vector<string>>, set<string>> first_alpha;
29:
30:vector<string> lefttest_increase;
31:stack <string> analysis_stack;

```

analyzer_table 是分析表，包括所有 Vt 符号的优先关系，同时，本人通过对文法的拓广，由原来的算法，直接获得了算术文法对#的优先关系。

firstVt 和 lastVt 是保存着所有 Vn 符号的相应集合。analysis_stack 为分析栈结构，在分析表驱动函数中用处颇多，lefttest_increase 为一个分析栈 pop 元素的记录结构，主要用来判断通过分析表和分析栈程序得到的短语是否为最左素短语，这将会和 grammar 的语法结构体 Vt、Vn 集之间相关联。

使用 map 结构获得由字符串结构作为类似 index 的表结构替代是实验 3 中的成果。这里使用 map 结构完成了 ppt 中讲授的以 Vt 或者 Vn 符号作为二维数组 index 的机器化。

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 firstVt 集合 lastVt 集，在这基础上构建出所有 Vt 的优先关系分析表。

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，语法分析部分的输出当然还是词法分析的结果，也就是二元式，同时本程序为了报错的方便，在二元式的基础上加入了行号 line，变成了三元式。其中的相应 config 配置依赖文件参考实验 1，包括各种文件依赖。

3.3.2. 函数定义

create_Vn_Vt_grammar(grammar_path); 创建 Vn Vt 集和文法表。
 create_firstVt(); 创建 firstVt 集。
 create_lastVt(); 创建 lastVt 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。

和课堂教授主要的区别在于分析栈函数。我们结合课堂讲授的程序流程图和基于 stl 的数据结构删去了 goto 语句，加入了标志符判断转移的来向，相对课堂上的 ppt 程序流程图，有所不同。

规约的产生式右部寻找与报错相对为原创。原理为暴力遍历。

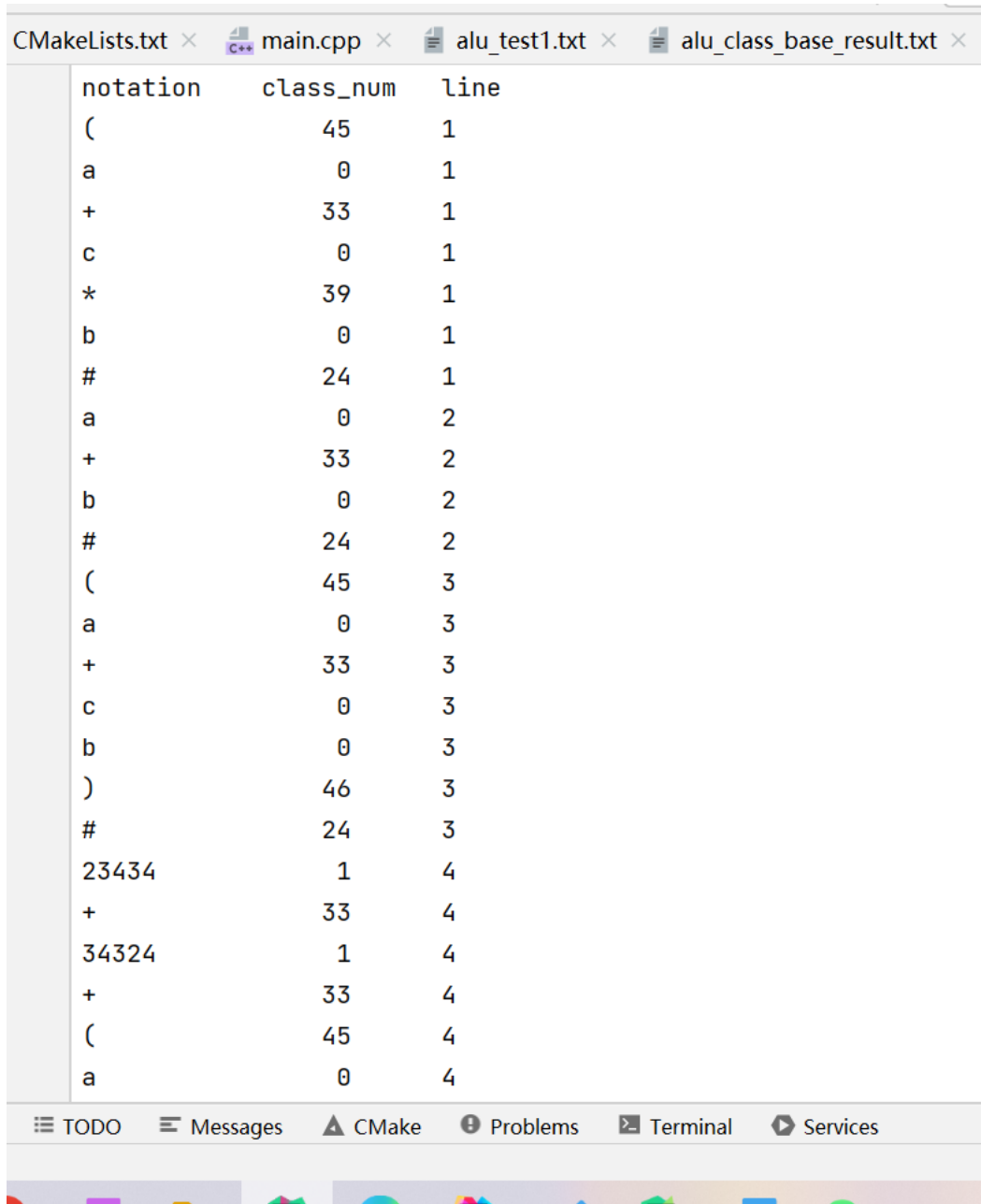
4. 程序测试

Alu_test1.txt

这里一个文件中包含了 6 个测试样例。

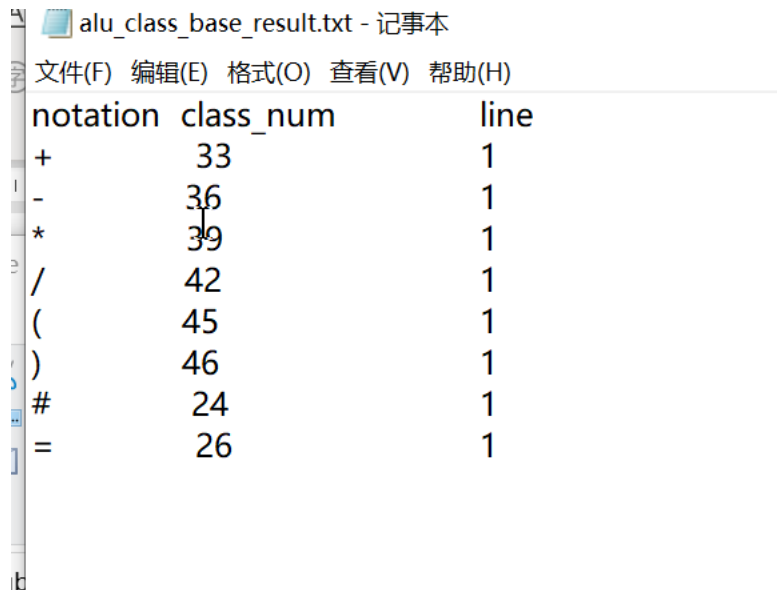
```
32:
33:(a+c * b #
34:a + b #
35:(a+c b ) #
36:23434+34324+(a* b / c) +(a*b/ (2-2)) #
37:( a+ b c ) * 32 / a #
38:a- b * #
```

由于是从词法分析开始，下边给出中间的结果的输出：



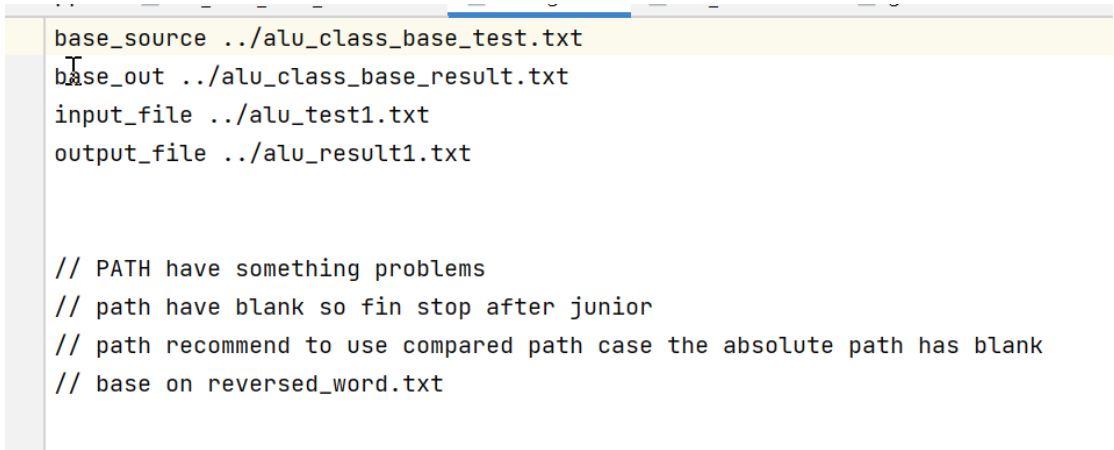
notation	class_num	line
(45	1
a	0	1
+	33	1
c	0	1
*	39	1
b	0	1
#	24	1
a	0	2
+	33	2
b	0	2
#	24	2
(45	3
a	0	3
+	33	3
c	0	3
b	0	3
)	46	3
#	24	3
23434	1	4
+	33	4
34324	1	4
+	33	4
(45	4
a	0	4

图表 4-1 test1 中词法分析结果



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 E T F
2  Vt + - * / = ( ) i n #
3
4  E' -> # E #
5  E -> E + T | E - T | T
6  T -> T * & | T / F | F
7  F -> ( E ) | i | n
8
9  end
10 // e is epilog
11 // each line ends with nothing,
12 // Vt i is identifier num
13 |

```

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 Principle\Experie
line 1 has syntax error
line 2 syntax correct
line 3 has syntax error
line 4 syntax correct
line 5 has syntax error
line 6 has syntax error

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。等式中，实验指导书原文法标识符位置，可以出现数字常量。这是对文法的拓展。

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。

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，这在算法表达式中 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：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。

$$\begin{aligned} E' &\rightarrow \# E \# \\ E &\rightarrow E + T \mid E - T \mid T \\ T &\rightarrow T * F \mid T / F \mid F \\ F &\rightarrow (E) \mid i \mid n \end{aligned}$$

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。

算法流程应该要和数据结构配合，课上讲解的算法流程图直接编写甚至 `goto` 语句，要原流程图分析得到算法原理，再结合新的数据结构给出新的算法流程图，再进行代码编写。

Clion debug 变量监视，如果在变量创建之前开始监视，会出现问题。