**PL/0编译器设计文档**

## 一、实验目的

### 1.1具体任务

文件输入：符合PL/0文法的源程序（自己要有5个测试用例，包含出错的情况，还要用老师提供的测试用例进行测试）

输出：P-Code

错误信息：参见教材第316页表14.4（新书第411页表17.4）。

错误信息尽量详细（行号，错误类型）

P-Code指令集：参见教材第351页表15.14。

语法分析部分要求统一使用递归下降子程序法实现。

编程语言自定，可使用C、C++、C#或Java等。（尽量选用有可执行文件的）过程。

解释程序

良好的GUI界面

### 1.2实验目标

《编译原理》课程设计是编译原理课程必不可少的一个环节，通过课程设计，加深对编译原理的教学内容的了解，以及实现编译原理各部分知识的融合。进而提高学生分析问题、解决问题，从而运用所学知识解决实际问题的能力。

在分析理解一个教学型编译程序（如PL/0）的基础上，对其词法分析程序、语法分析程序和语义处理程序进行实现，达到进一步了解程序编译过程的基本原理和基本实现方法的目的。

## 二、PL/0编译器的设计分析

### 2.1 PL/0文法概述

PL/0语言是一种类PASCAL语言，是教学用程序设计语言，它比PASCAL语言简单，作了一些限制。PL/0的程序结构比较完全，赋值语句作为基本结构，构造概念有：

- 顺序执行、条件执行和重复执行，分别由begin/end,if then else和while do语句表示。

- PL0还具有子程序概念，包括过程说明和过程调用语句。

- 在数据类型方面，PL0只包含唯一的整型，可以说明这种类型的常量和变量。

- 运算符有+，-，\*，/，=，<>，<，>，<=，>=，(，)。

- 说明部分包括常量说明、变量说明和过程说明

### 2.2 PL/0文法的EBNF表示

<程序> ::= <分程序>.

<分程序> ::= [<常量说明部分>][变量说明部分>][<过程说明部分>]<语句>

<常量说明部分> ::= const<常量定义>{,<常量定义>};

<常量定义> ::= <标识符>=<无符号整数>

<无符号整数> ::= <数字>{<数字>}

<标识符> ::= <字母>{<字母>|<数字>}

<变量说明部分>::= var<标识符>{,<标识符>};

<过程说明部分> ::= <过程首部><分程序>{<过程说明部分>};

<过程首部> ::= procedure<标识符>;

<语句> ::= <赋值语句>|<条件语句>|<当型循环语句>|<过程调用语句>|<读语句>|<写语句>|<复合语句>|<重复语句>|<空>

<赋值语句> ::= <标识符>:=<表达式>

<表达式> ::= [+|-]<项>{<加法运算符><项>}

<项> ::= <因子>{<乘法运算符><因子>}

<因子> ::= <标识符>|<无符号整数>|'('<表达式>')‘

<加法运算符> ::= +|-

<乘法运算符> ::= \*|/

<条件> ::= <表达式><关系运算符><表达式>|odd<表达式>

<关系运算符> ::= =|<>|<|<=|>|>=

<条件语句> ::= if<条件>then<语句>[else<语句>]

<当型循环语句> ::= while<条件>do<语句>

<过程调用语句> ::= call<标识符>

<复合语句> ::= begin<语句>{;<语句>}end

<重复语句> ::= repeat<语句>{;<语句>}until<条件>

<读语句> ::= read'('<标识符>{,<标识符>}')‘

<写语句> ::= write'('<标识符>{,<标识符>}')‘

<字母> ::= a|b|...|X|Y|Z

<数字> ::= 0|1|2|...|8|9

Tips：

数据类型：无符号整数

标识符类型：简单变量(var)和常数(const)

数字位数：小于14位

标识符的有效长度：小于10位

过程嵌套：小于3层

### 2.3 PL/0编译器的系统结构

PL/0编译系统是一个编译-解释执行程序,整个编译过程分两个阶段进行。第一阶段先把PL/0源程序编译成假想计算机的目标(P-code指令)程序,第二阶段再对该目标程序进行解释执行,得到运行结果。PL/0编译程序采用- -遍扫描，即以语法分析为核心,由它调用词法分析程序取单词,在语法分析过程中同时进行语义分析处理,并生成目标指令。如遇语法、语义错误,则随时调用出错处理程序,打印出错信息。在编译过程中要利用符号表的登录和查找来进行信息之间的联系。一遍扫描的PL/0编译和P-code解释执行框图如图1所示：

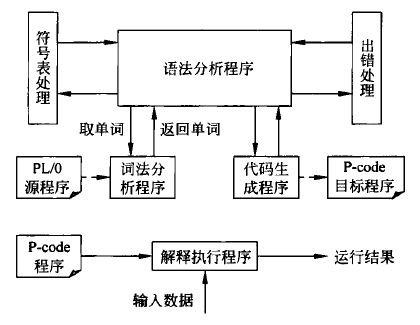


图1 PL/0编译系统结构框图

## 三、实验步骤

**2.4构建算符优先矩阵——priority\_array（）**

只是构建出了上面的两种集合是不够的，由上面分析我们知道了这两个集合的用处。说白了就是为了构造出算符优先分析表。关系分为四类1.等于，2.大于，3.小于，4没有关系。

     实现getPriority\_array()的伪代码如下：

|  |
| --- |
| FOR 每个产生式 P->X1 X2 ……Xn  DO FOR i:=1 TO n-1 DO  IF X[i]和X[i+1]均为终结符  THEN 置 X[i]=X[i+1]  IF X[i]和X[i+2]均为终结符,X[i+1]为非终结符,i≤n-2,  THEN 置 X[i]=X[i+2]  IF X[i]为终结符, 但X[i+1]为非终结符  THEN FOR FIRSTVT(X[i+1])中的每个a  DO 置 X[i]<a  IF Xi为非终结符, 但X i+1 为终结符  THEN FOR LASTVT(X i )中的每个a  DO 置 a>X[i+1] |

**2.5进行算符优先分析——PriorityAnalysis（）**

步骤如下：

1.初始化：#入栈，在输入串尾部加上#，栈顶指针p，输入串头指针为s

2.p指向的符号为非终结符则下移，为终结符(记为a)则执行步骤3

3.s指向的符号记为b，查算符优先分析表，若(a,b)为<或者=，则将s指向的符号入栈，若为>则执行步骤4

4.从栈顶开始向下找两个终结符c,d（c靠近栈顶，c和d之间不存在其它终结符，c和d相邻或者其间只有非终结符）,查表若(d,c)为=，则继续向下找终结符e,将d,e重新记为c,d，重复步骤4，若(d,c)为<则执行步骤5，若为空则不匹配，输入串不是该文法的句子

5.d上面到栈顶的所有符号为最左素短语，可规约为非终结符，栈顶指针p指向这个非终结符，s指向下一个符号，栈中只存在一个非终结符和#，输入串只剩下#，则规约完毕输入串是该文法的句子，否则执行步骤2.

6.规约时将非终结符均规约为‘E’

**2.5 算符分析器的关键实现代码与数据结构**

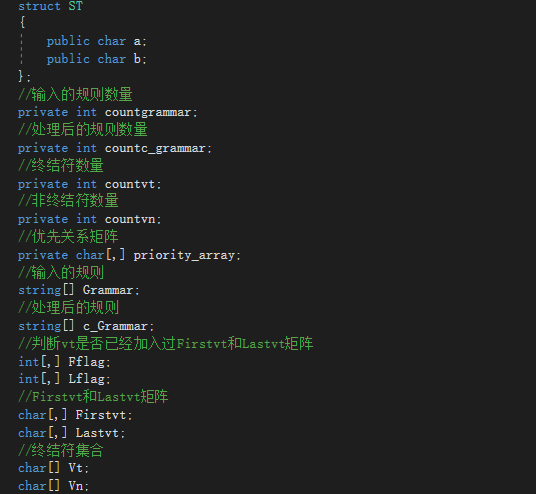


图3使用的数据结构



图4 实现的主要方法

1. **算符优先分析器**

**3.1实际运行截图**

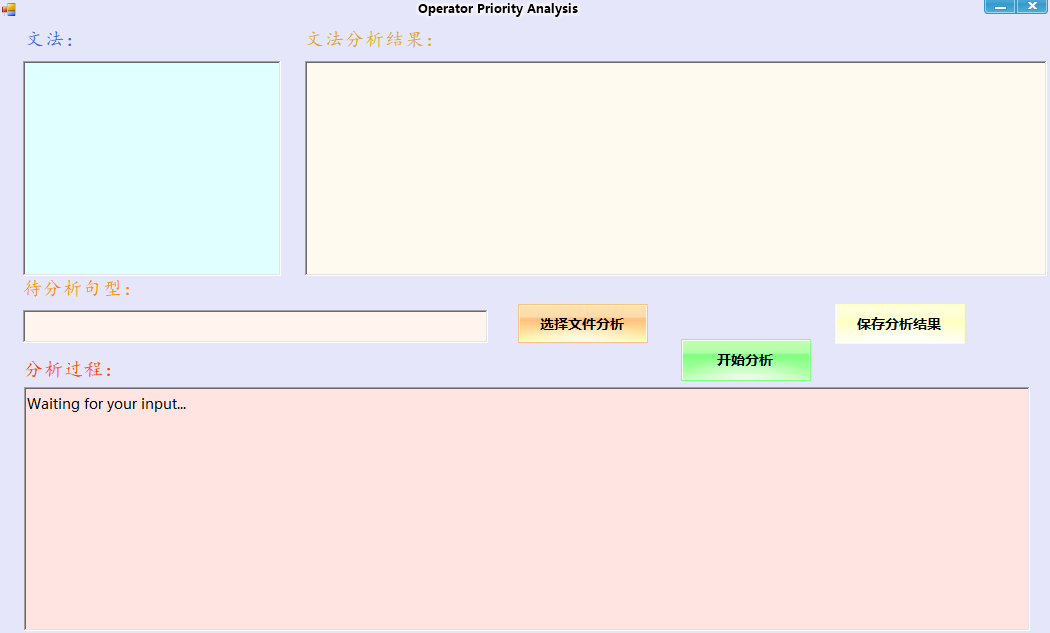


图5 运行前截图



图6 运行时截图

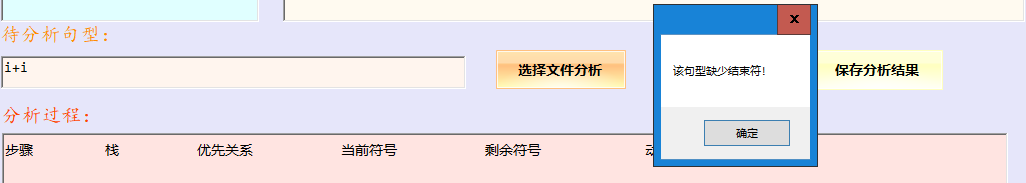


图7错误报告-1

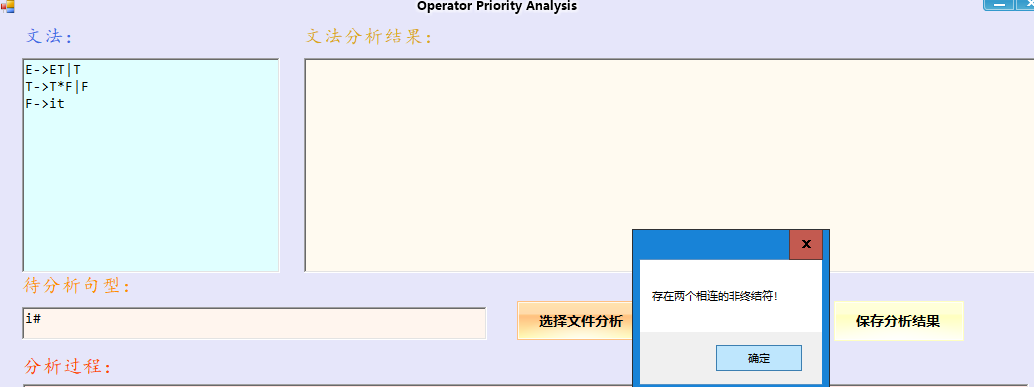


图8错误报告-2

通过【选择文件分析】按钮可以查找并导入计算机中的代码文件于【文法区】，也可以直接在【文法区修改】或添加文法；添加【待分析句型】后，通过点击【开始分析】按钮可以进行算法优先分析，将结果输出于【分析过程区】（文法分析的中间结果将显示在【文法分析结果区】），最后可以通过【保存分析结果】将【文法分析结果区】中的结果保存为文本文件于指定地址。

本算符优先分析器对输入的文法还做出了限制：Vn为介于‘A’~‘Z’的字母，并且规约过程中非终结符都规约为‘E’，规则中需要有形如“S->#E#”的规则。

**四、实验感想**

通过此次实验，使我更加扎实的掌握了编译过程中算符优先分析方面的知识，尽管在之前的课堂中了解了算符优先文法的实现思路，但是理论的学习与实际上的编写程序过程相比还是有一些差距(比如算法)，经过本次实验，尽管在设计过程中还是遇到了一些问题，但经过一次又一次的思考，一遍又一遍的检查终于找出了原因所在，也暴露出了之前我在这方面的知识仍然存在欠缺和经验不足，课下通过和同学之间的交流，让我对于针对算符优先文法的可能存在的错误以及处理方式有了更加深入的理解，尽管设计还是存在一些遗憾（只能针对有限制的算符优先文法），这些都是在之前的设计过程中欠缺考虑的一些问题，通过本次实验对于算符优先分析器的编写，使我对于算符优先分析的流程有了更加清晰的认识，而且在程序代码的编写以及可视化界面的设计上也是对上个学期C#知识的回顾与应用；实践出真知，通过这次课程设计使我懂得了理论与实际相结合是很重要的，只有理论知识是远远不够的，只有把所学的理论知识与实践相结合起来，从理论中得出结论，才能提高自己的实际动手能力和独立思考的能力，通过亲自动手制作，使我掌握的知识不再是纸上谈兵，受益匪浅。