



**课程设计**

课程名称 编译原理

题目名称 PL/0编译器的扩充

学生学院 计算机学院

专业班级 计算机科学与技术15(1)

学 号 3115004889

学生姓名 骆栋浩

指导教师 李杨

|  |  |
| --- | --- |
| 程序功能完成情况 |  |
| 测试用例全面程度 |  |
| 学生对所编程序熟悉程度 |  |
| 报告格式是否与要求相符 |  |
| 报告内容是否准确、全面 |  |

2018年 7月 2日

目录

[PL/0编译器的扩充](#_Toc29948_WPSOffice_Level1) [3](#_Toc29948_WPSOffice_Level1)

[一、 实验内容](#_Toc28114_WPSOffice_Level1) [3](#_Toc28114_WPSOffice_Level1)

[二、 实现内容](#_Toc27237_WPSOffice_Level1) [3](#_Toc27237_WPSOffice_Level1)

[三、 结构设计说明](#_Toc21286_WPSOffice_Level1) [4](#_Toc21286_WPSOffice_Level1)

[四、 主要成分描述](#_Toc11296_WPSOffice_Level1) [5](#_Toc11296_WPSOffice_Level1)

[五、 开发过程](#_Toc7022_WPSOffice_Level1) [7](#_Toc7022_WPSOffice_Level1)

[六、 测试用例和结果](#_Toc771_WPSOffice_Level1) [12](#_Toc771_WPSOffice_Level1)

[七、 实验感想](#_Toc32569_WPSOffice_Level1) [14](#_Toc32569_WPSOffice_Level1)

# PL/0编译器的扩充

### 实验内容

基本内容

1. 扩充赋值运算：\*= 和 /=
2. 扩充语句（Pascal的FOR语句）:

FOR <变量>:=<表达式>STEP<表达式> UNTIL<表达式>Do<语句>

选做内容

1. 增加类型：① 字符类型； ② 实数类型。
2. 扩充运算：++ 和 --（要求作为表达式实现）
3. 扩充函数：① 有返回值和返回语句；② 有参数函数。
4. 增加一维数组类型（可增加指令）。
5. 其他典型语言设施。

### 实现内容

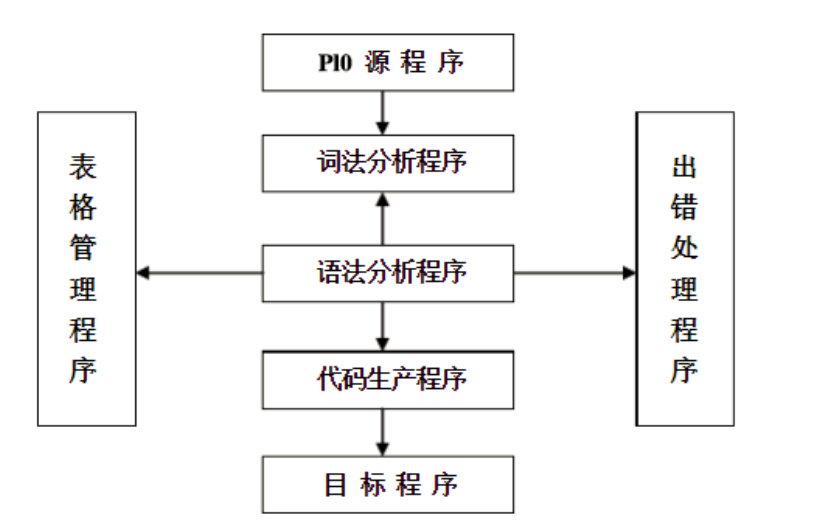
1. 扩充赋值运算：\*= 和 /=
2. 扩充语句（Pascal的FOR语句）:

FOR <变量>:=<表达式>STEP<表达式> UNTIL<表达式>Do<语句>

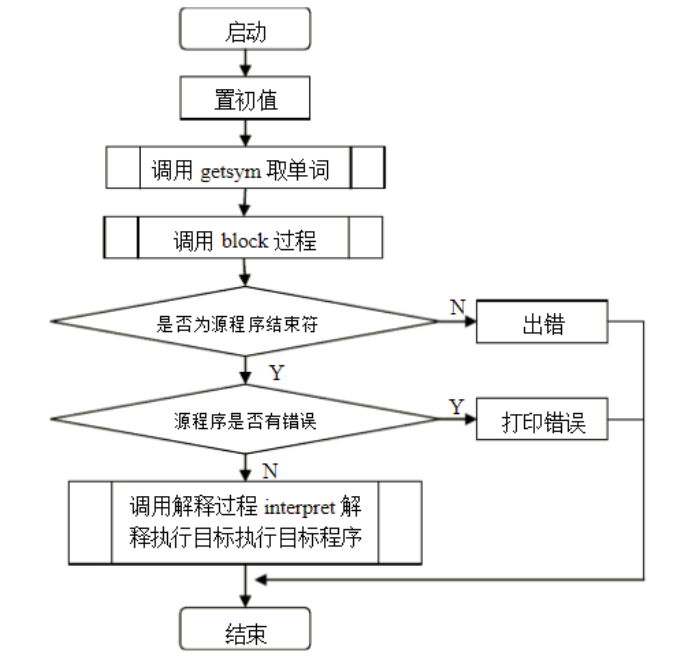
1. 扩充运算：++ 和 --（要求作为表达式实现）

### 结构设计说明

3.1 PL/编译程序结构图



3.2 PL/0编译程序总体流程图



3.3 各功能模块描述

1. SymIn（）：判断SYMBOL是否属于集合SYMSET
2. SymSetUnion():求两个SYMBOL的交集
3. SymSetadd()：在SYMSET中添加元素添加SYMBOL
4. SymSetNew()：构造一个SYMSET
5. SymSetBULL()：构造一个空的SYMSET
6. Error():出错处理
7. GetCh():PL0文件中读取字符
8. GetSym():识别一个单词
9. GEN():目标代码生成
10. TEST():测试单词是否合法
11. ENTER():在表中加入一个object
12. POSITION():查找名字的位置
13. ConstDeclaration(): 常量声明处理
14. VarDeclaration():变量声明处理
15. ListCode():打印PL0代码
16. TERM():项处理
17. EXPRESSION():表达式处理过程
18. CONDITION():条件处理过程
19. STATEMENT():语句处理
20. Block():编译程序块
21. Interpret():执行程序

### 主要成分描述

### 4.1 符号表

为了组成一条指令，编译程序必领知道其操作码及其参数（数或地址）。 这些值是由编译程序本身联系到相应标识符上去的。这种联系是处理常数、变量和过程说明完成的。为此，标识符表应包含每一标识符所联系的属性；如果标识符被说明为常数，其属性值为常数值；如果标识符被说明成变量，其属性就是由层次和修正量（偏移量）组成的地址；如果标识符被说明为过程，其属性就是过程的入口地址及层次。 常数的值由程序正文提供，编译的任务就是确定存放该值的地址。 我们选择顺序分配变量和代码的方法;每遇到一个变量说明， 就将数据单元的下标加一（ PL/0 机中，每个变量占一个存储单元）。 开始编译一个过程时， 要对数据单元的下标 dx 赋初值，表示新开辟一个数据区。 dx 的初值为 3,因为每个数据区包含三个内部变量 RA， DL 和 SL。

**4.2 运行时存储组织和管理**

对于源程序的每一个过程（包括主程序），在被调用时， 首先在数据段中开辟三个空间，存放静态链 SL、动态链 DL 和返回地址 RA。 静态链记录了定义该过程的直接外过程（或主程序）运行时最新数据段的基地址。动态链记录调用该过程前正在运行的过程的数据段基址。返回地址记录了调用该过程时程序运行的断点位置。 对于主程序来说， SL、 DL 和 RA 的值均置为 0。 静态链的功能是在一个子过程要引用它的直接或间接父过程（这里的父过程是按定义过程时的嵌套情況来定的，而不是按执行时的调用顺序定的）的变量时，可以通过静态链， 跳过个数为层差的数据段，找到包含要引用的变量所在的数据段基址，然后通过偏移地址访问它。

在过程返回时， 解释程序通过返回地址恢复指令指针的值到调用前的地址，通过当前段基址恢复数据段分配指针，通过动态链恢复局部段基址指针。 实现子过程的返回。 对于主程序来说， 解释程序会遇到返回地址为 0 的情況，这时就认为程序运行结束。

解释程序过程中的 base 函数的功能， 就是用于沿著静态链，向前查找相差指定层数的局部数据段基址。 这在使用 sto、 lod、 stoArr、 lodArr 等访问局部变量的指令中会经常用到。

类 PCODE 代码解释执行的部分通过循环和简单的 case 判断不同的指令 0，做出相应的动作。 当遇到主程序中的返回指令时，指令指针会指到 0 位置，把这样一个条件作为终至循环的条件，保证程序运行可以正常的结束。

**4.3 语法分析方法**

语法分析子程序采用了自顶向下的递归子程序法，语法分析同时也根据程序的语义生成相应三元代码，并提供了出错处埋的机制。 语法分析主要由分程序分析过程（ BLOCK)、参数变量分析过程（ ParaDeclaraction)、参数变量处理过程（ ParaGetSub)、数组处理过程(ParaGetSub)、 常量定义分析过程（ ConstDeclaration)、 变量定义分析过程(Vardeclaration)、语句分析过程（ Statement）、表达式处理过程（ Expression）、 项处埋过程（ Term)、因子处埋过程（ Factor)和条件处理过程（ Condition)构成。 这些过程在结构上构成一个嵌套的层次结构。 除此之外，还有出错报告过程（ Error)、代码生成过程（ Gen)、测试单词合法性及出错恢复过程（ Test)、登录名字表过程（ Enter)、查询名字表函数（ Position)以及列出类 PCODE代码过程（ Listcode)作过语法分析的辅助过程。

**4.4 代码生成**

中间代码是是源程序的一种内部表示， 复杂性介于源语言和目标机语言之间。

中间代码的表示方法有逆波兰式、三元式、树形、四元式等。

1) 逆波兰记号是最简单的一种中间代码表示形式，早在编译程序出现之前，它就用于表示算术表达式。后缀表示法表示表达式，其最大的优点是易于栈式计算机处埋表达式。

2) 每个三元式由三个部分组成：

A． 算符 op

B． 第一运算对象 ARG1

C． 第二运算对象 ARG2

运算对象可能是源程序中的变量，也可能是某个三元式的结果，用三元式的编号表示。

3) 树形表示是三元式表示的翻版。

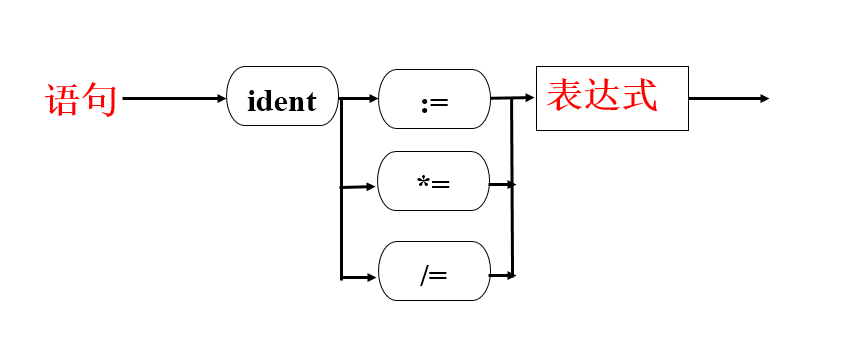
4) 四元式是一种比较普遍采用的中间代码形式：

算符 op，运算对象 ARS1，运算对象 ARG2， 运算结果 RESULT。

### 开发过程

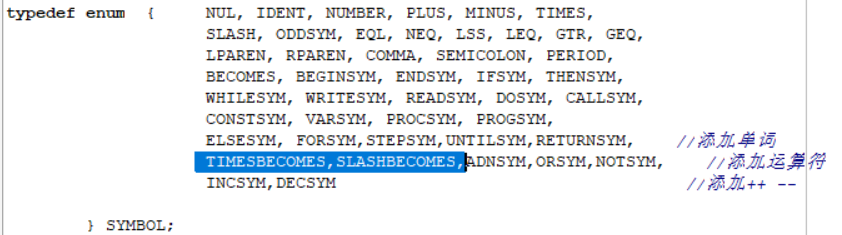
5.1 扩充赋值运算\*=,/=

5.1.1 语法描述图

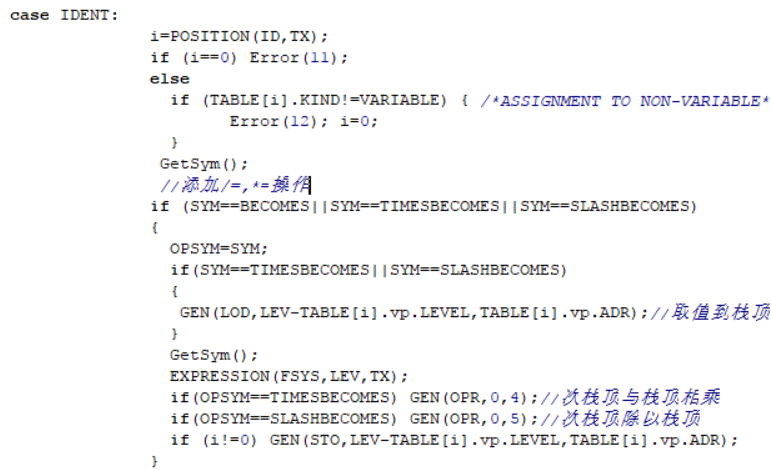


5.1.2 代码实现

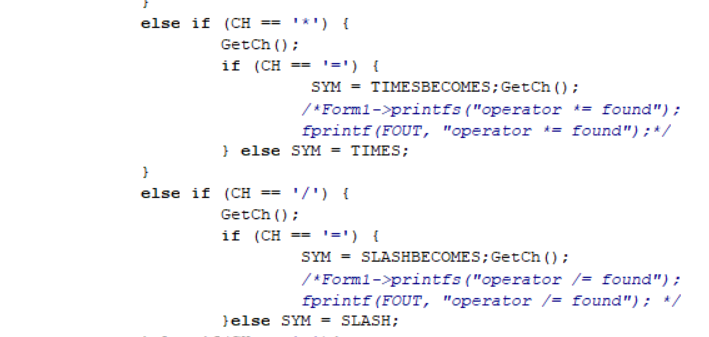
1. 添加运算符关键字



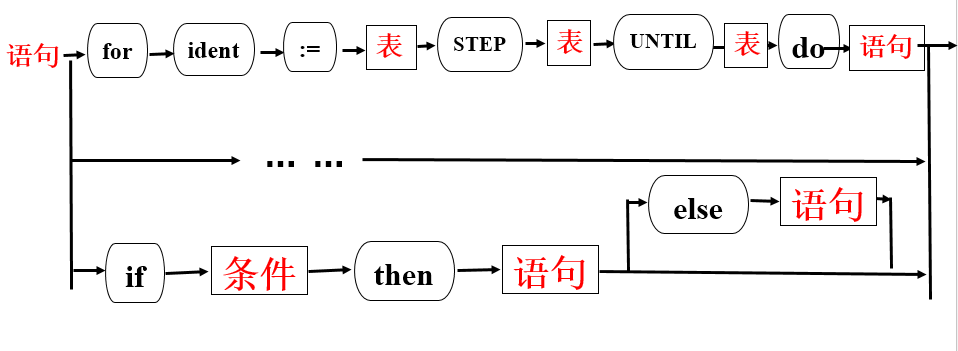
1. 在STATEMENT()函数中添加 \*= 、 /= 操作



1. 修改GetCh()函数，增加对 \*= 、 /=的检测

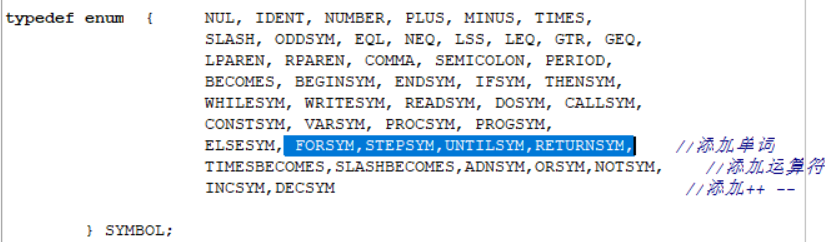
5.2 扩充语句（Pascal的FOR语句）:

5.2.1 语法描述图

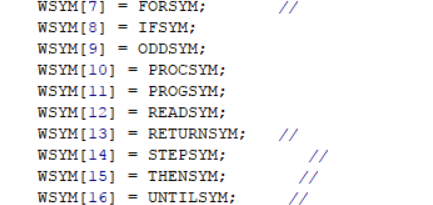
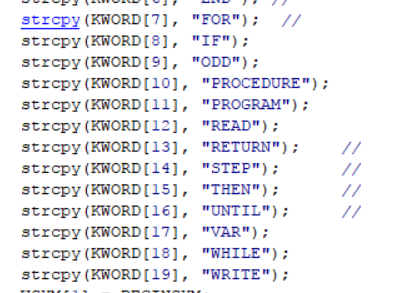


5.2.2 代码实现

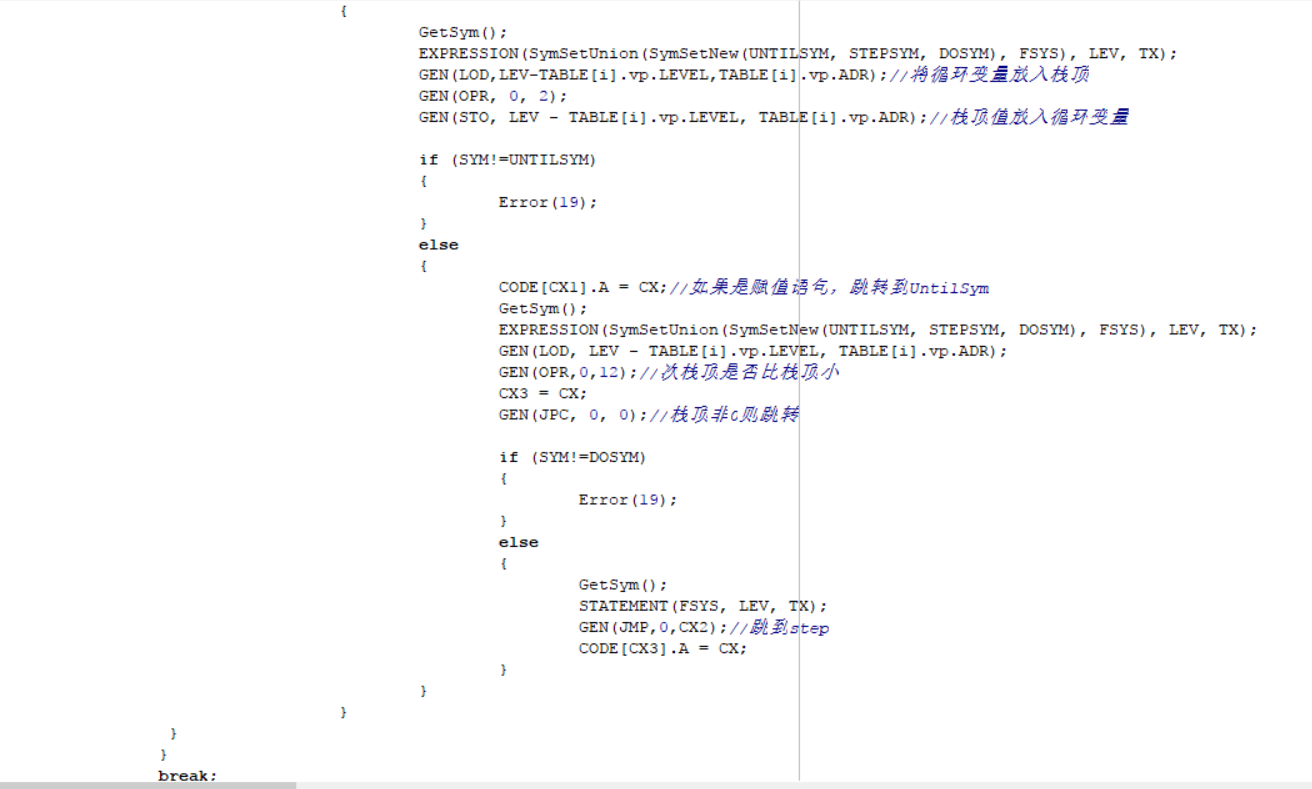
1. 添加关键字



1. 修改保留字数组



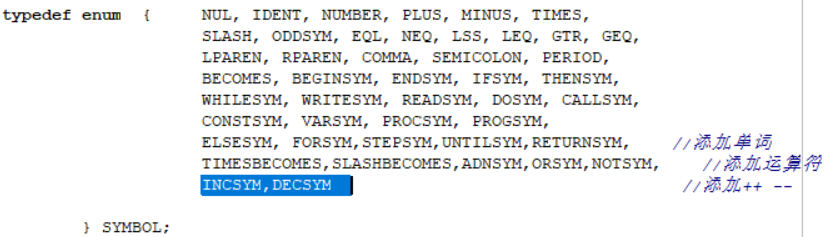
1. 在STATEMENT()函数中添加case FORSYM



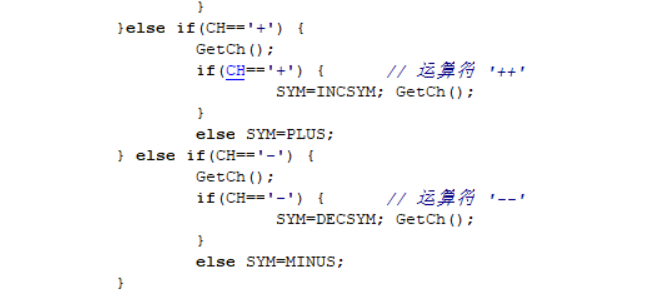
5.3 扩充运算：++ 和 --（要求作为表达式实现）

5.3.1 代码实现

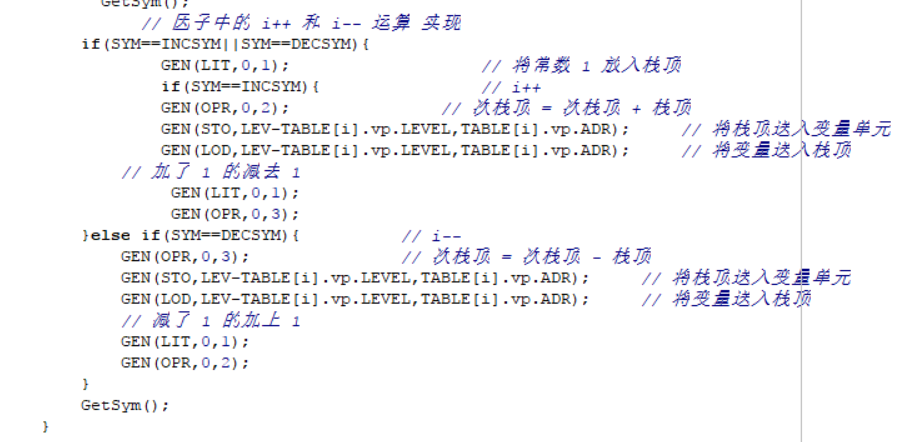
1. 添加关键字



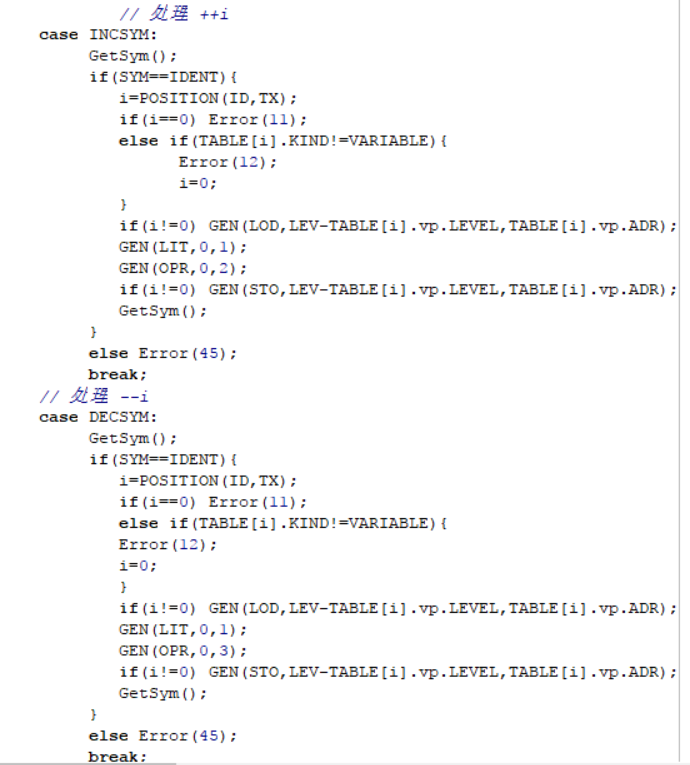
1. 修改GetCh()



1. 修改FACTOR()函数



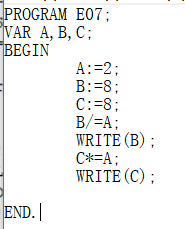
1. 在STATEMENT()函数中添加case INCSYM,case DECSYM



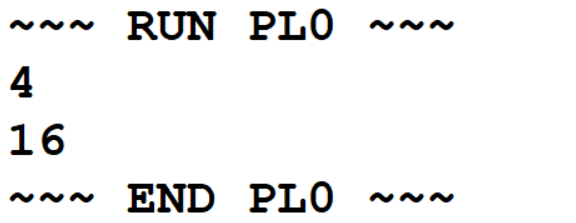
### 测试用例和结果

6.1 扩充赋值运算\*=,/=

6.1.1 测试用例 E07.PL0文件

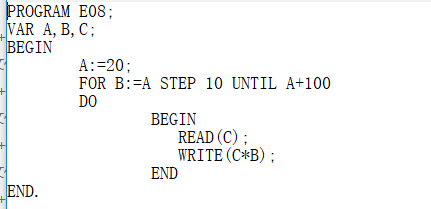


6.1.2 测试结果

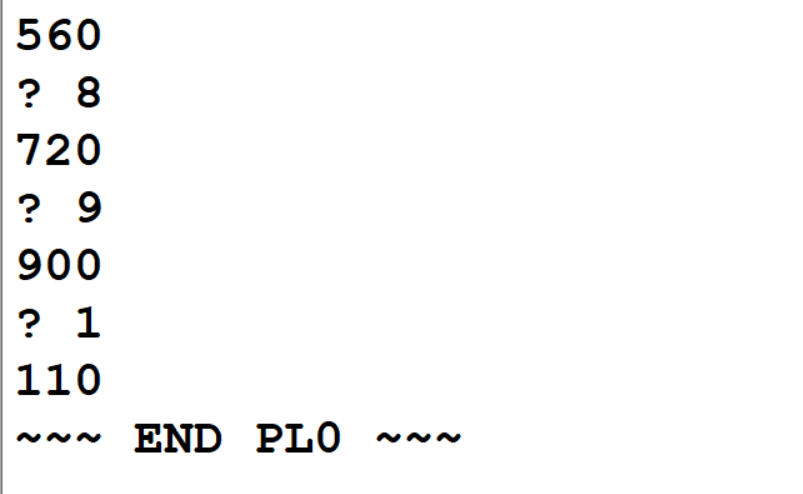
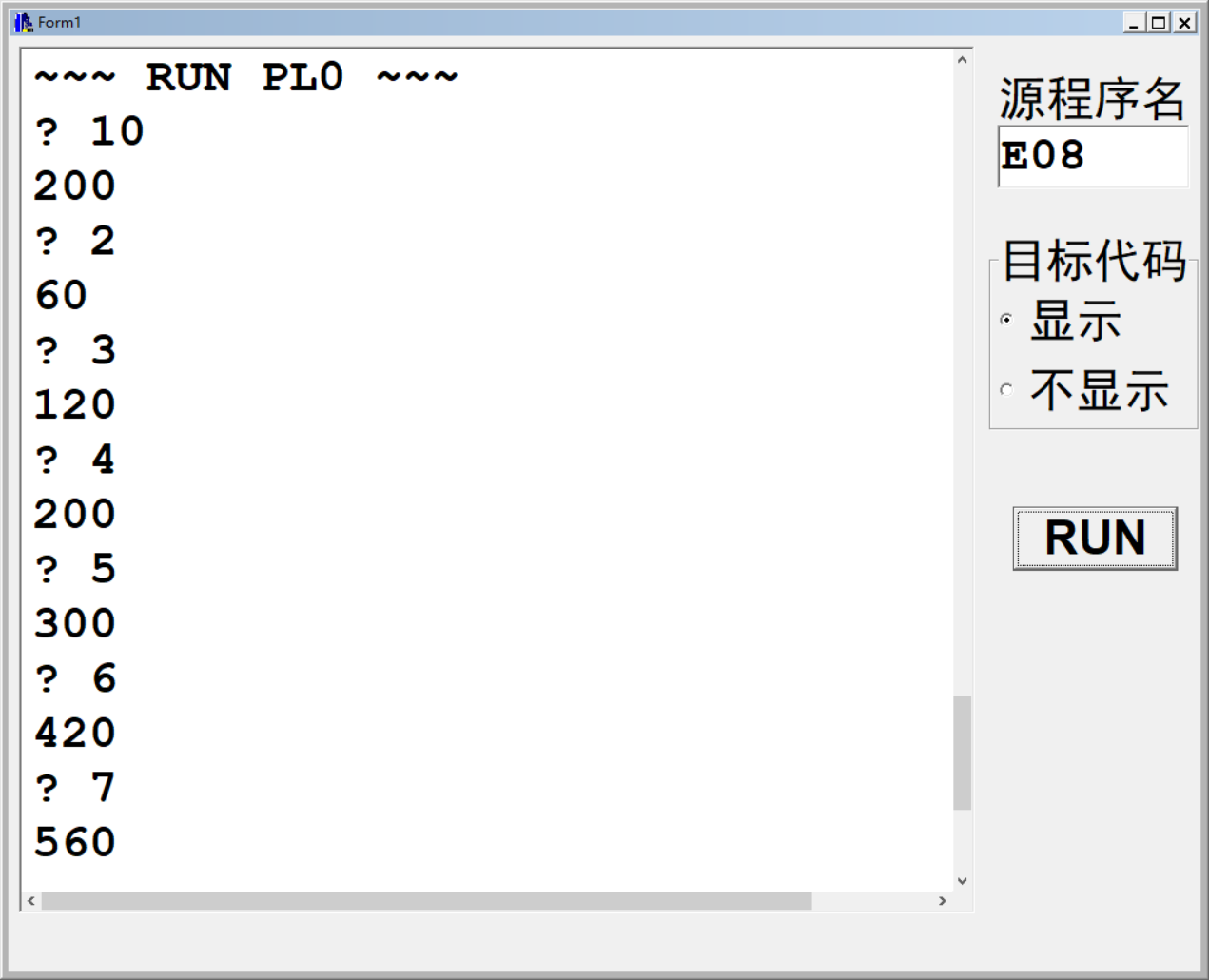


6.2 扩充语句（Pascal的FOR语句）:

6.2.1 测试用例 E08.PL0

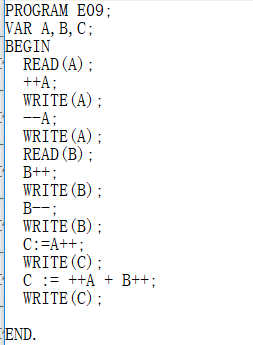


6.2.2 测试结果

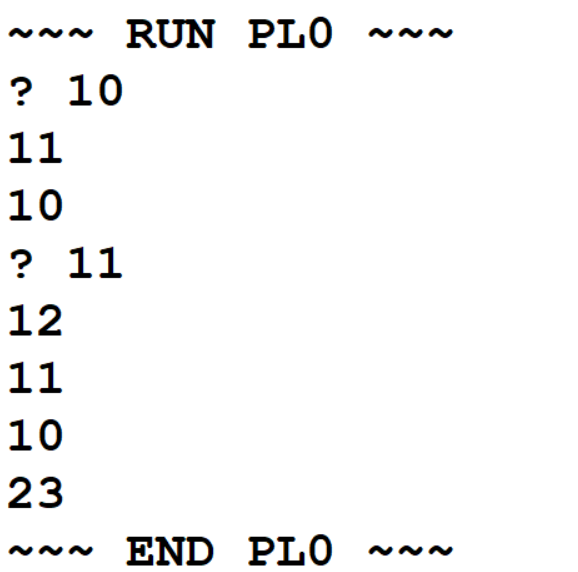


6.3扩充运算：++ 和 --（要求作为表达式实现）

6.3.1 测试用例 E09.PL0



6.3.2 测试结果



### 实验感想

本次课程设计主要是增添保留字和运算符以及修改单词是学习编译程序的基本操作，再加上前面有实验的一些基础，总体来说，没有之前做实验那么辛苦。但是在本次实验中，还是遇到了不少的问题，例如第一个功能的实现的时候，在实现/=功能的时候，一直打印的结果都是0，一直都很疑惑，到最后面才发，一直在执行的是取余操作，最后把取余操作改成除操作，结果就正确了。

通过这次实验，虽然自己对编译原理的知识有了进一步的了解和理解，但是发现自己还是有很多知识点没有掌握，后期还需要不断的复习和实训去掌握知识点。掌握知识点，才能更好的进行实验的操作，才能让效率更好，同时也能为自己以后写代码的时候有一定的帮助。