|  |  |  |
| --- | --- | --- |
| 评阅教师 | 设计成绩 | 评阅日期 |
|  |  |  |

**海南大学计算机与网络空间安全学院**

**编译原理课程设计报告**

****

**班 级：**

**成 员：**

**指导老师：**

**完成日期：**

目录：

[一、课程设计目的和要求 3](#_Toc45137517)

[二、程序的组织结构 4](#_Toc45137518)

[（一）程序结构图 4](#_Toc45137519)

[（二）模块成员和函数组成 5](#_Toc45137520)

[（三）编译器参数 6](#_Toc45137521)

[三、 程序流程图 7](#_Toc45137522)

[（一）程序总体流程图 7](#_Toc45137523)

[（二）语法分析流程图 8](#_Toc45137524)

[（三）IF分析流程图 9](#_Toc45137525)

[（四）do while流程图 9](#_Toc45137526)

[（五）For流程图 10](#_Toc45137527)

[四、系统实现 11](#_Toc45137528)

[五、程序源代码 14](#_Toc45137529)

[1、if else语句 14](#_Toc45137530)

[2、do while until语句 15](#_Toc45137531)

[3、for语句 16](#_Toc45137532)

[六、系统测试 19](#_Toc45137533)

[1、if else语句 19](#_Toc45137534)

[2、do while until语句 20](#_Toc45137535)

[3、for语句 22](#_Toc45137536)

[七、设计总结 24](#_Toc45137537)

[八、课题成员及分工 24](#_Toc45137538)

[九、自评成绩 24](#_Toc45137539)

## 课程设计目的和要求

对PL/0语言进行扩充。

**课程设计要求1：**

1）扩充语言成份：“if  条件  then  语句系列1  else  语句系列2”

2）写出相应的编译程序

**课程设计要求2：**

1）扩充语言成份：“do while

语句系列

until 条件”

意即循环执行循环体内的语句系列，直到条件为真为止。

2）写出相应的编译程序

**课程设计要求3：**

1）扩充语言成份：

①“for 变量= 初值  to  终值  do  begin  语句系列  end”

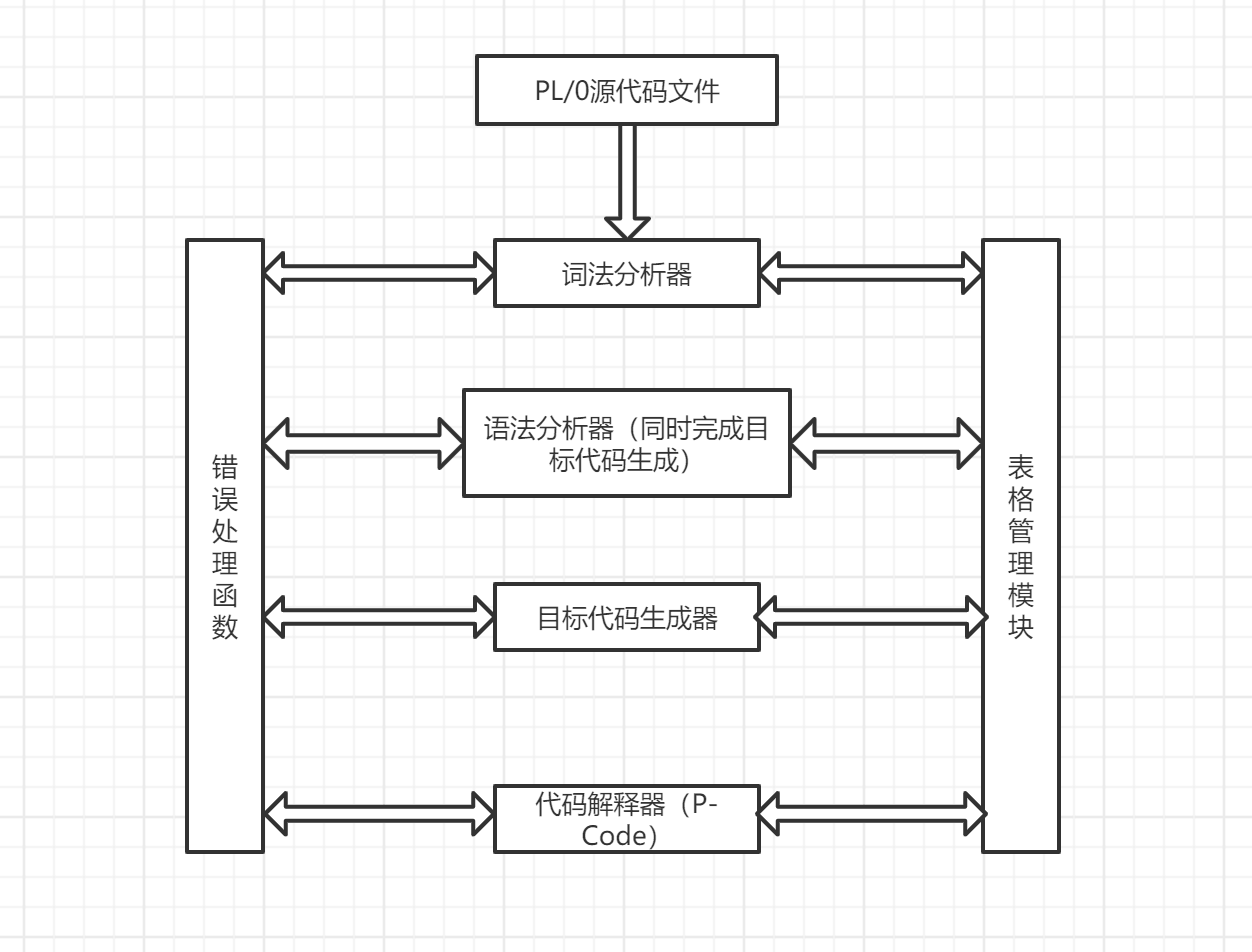
②“for 变量= 初值  downto  终值  do  begin  语句系列  end”

其中，语句①中循环变量的步长为1，语句②中循环变量的步长为-1。

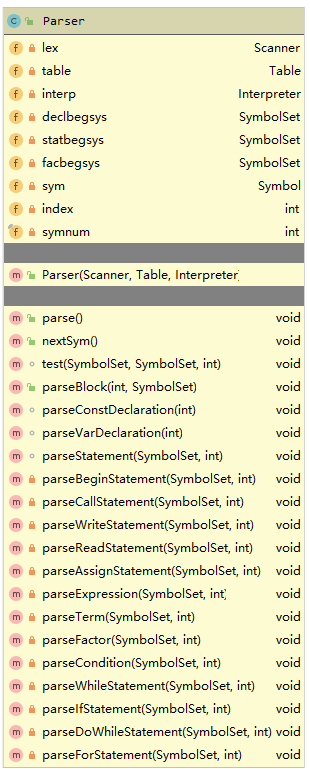
2）写出相应的编译程序

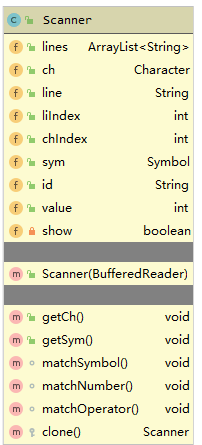
## 程序的组织结构

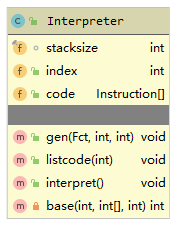
#### （一）程序结构图



#### （二）模块成员和函数组成





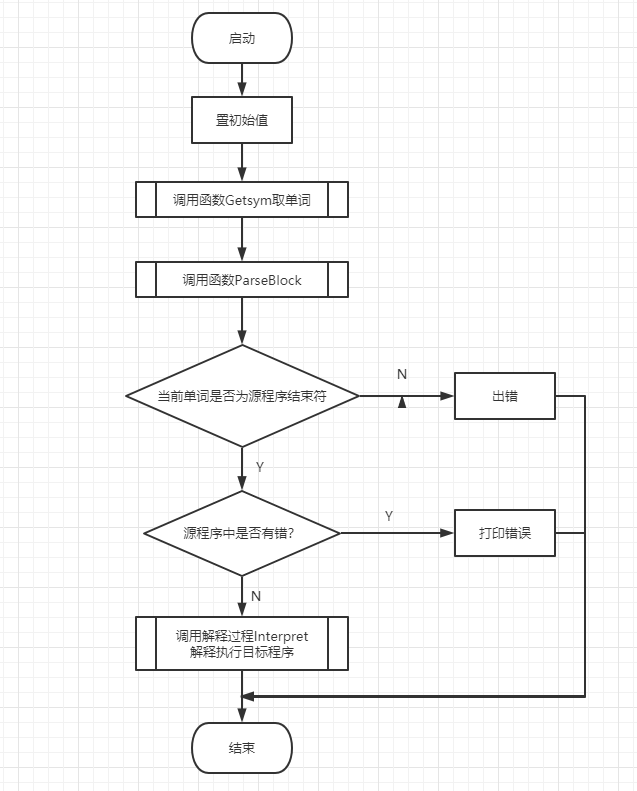


#### （三）编译器参数

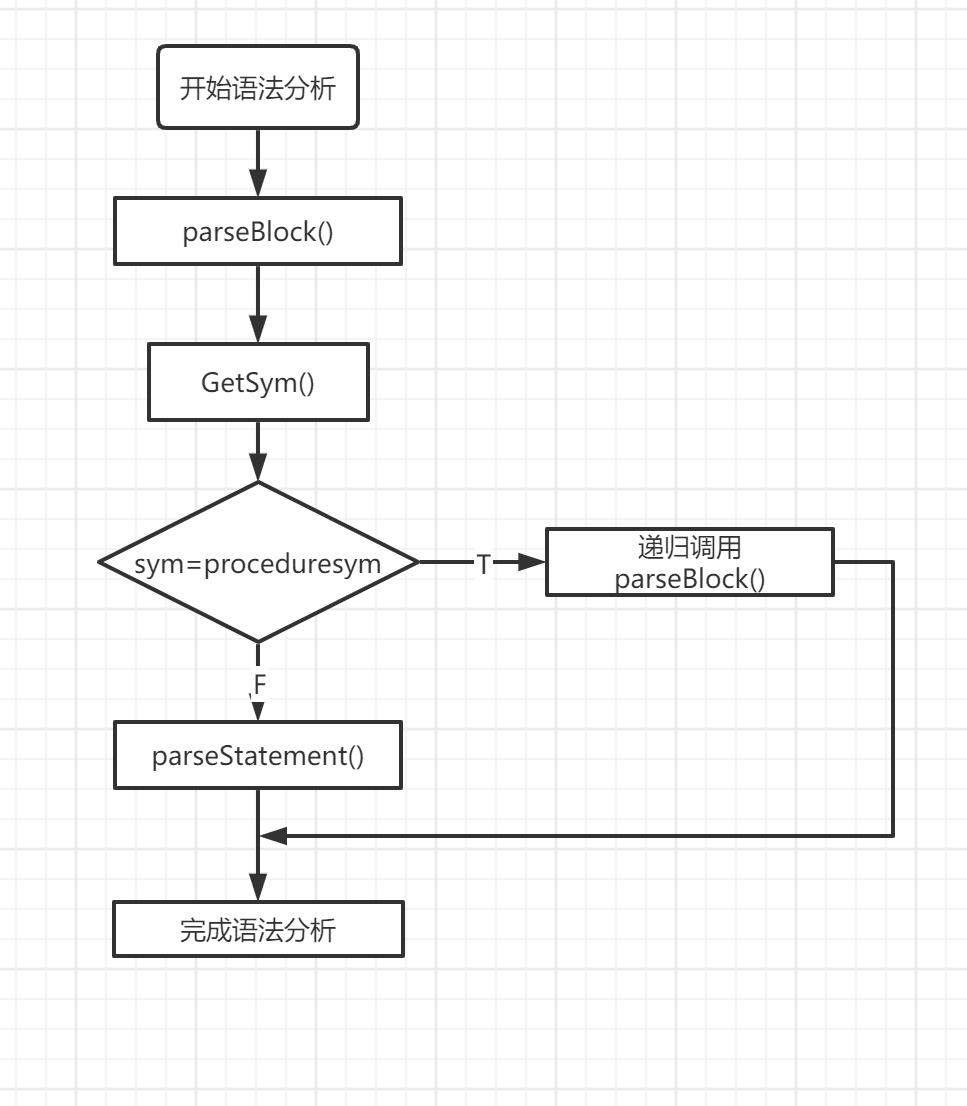
|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 默认值 | 意义 |
| MAX\_VIRTUAL\_CODE | Int | 500 | 最多虚拟机代码数 |
| MAX\_NUM | Int | 2048 | 数字最大值 |
| MAX\_DIGIT | Int | 14 | 数字最大位数 |
| MAX\_TABLE | Int | 100 | 名称表最大大小 |
| SHOW\_CODE | Boolean | True | 是否显示目标代码 |
| SHOW\_TABLE | Boolean | True | 是否显示名称表 |
| SHOW\_PL0CODE | Boolean | True | 是否显示原代码 |
| ERROR\_TEXT | String[] |  | 错误信息 |

## 程序流程图

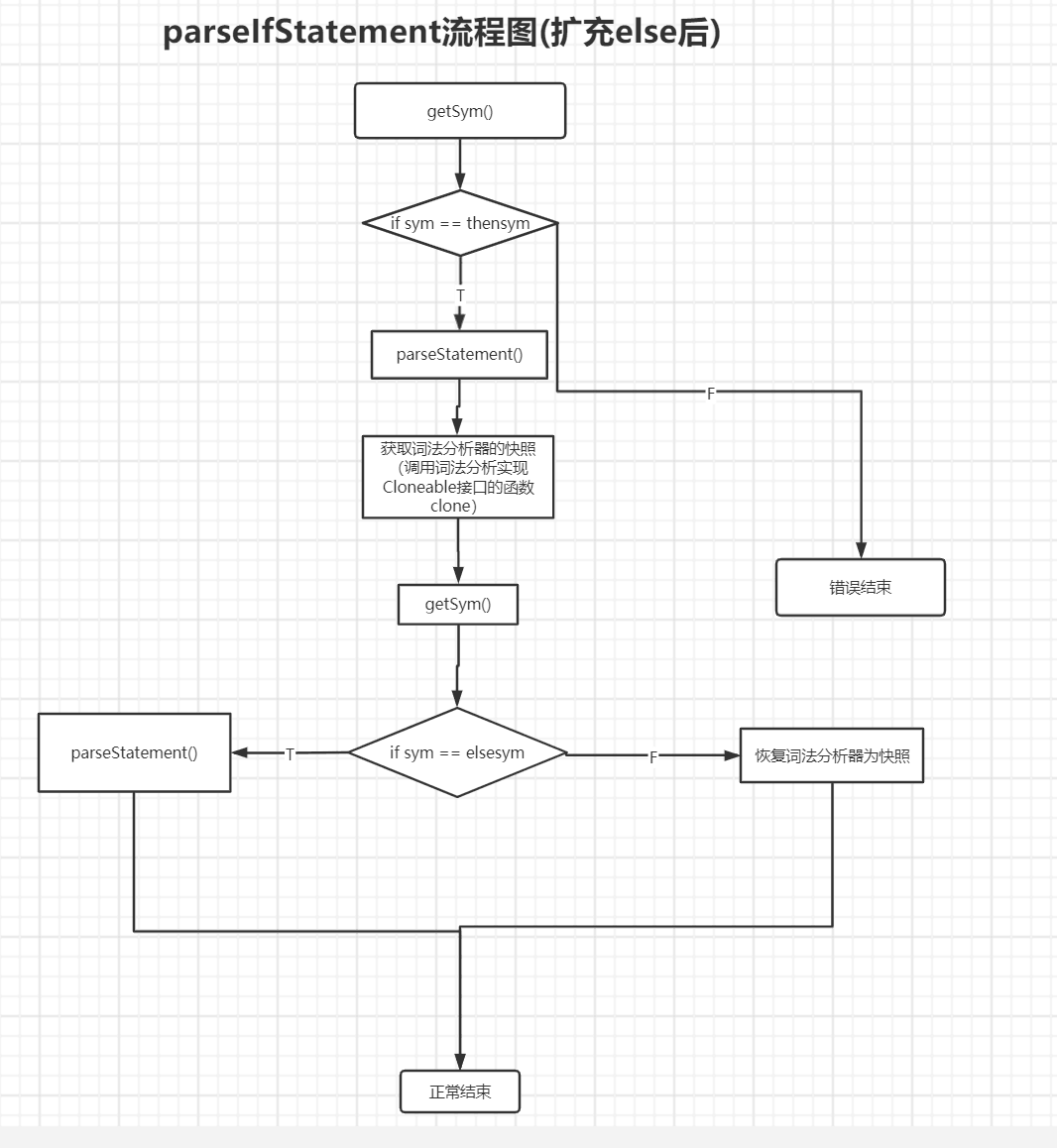
#### （一）程序总体流程图



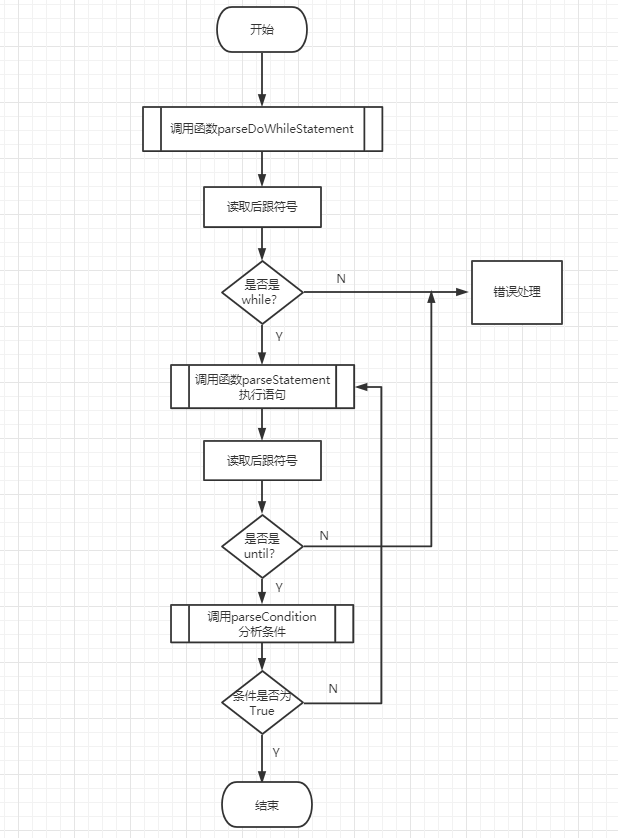
#### （二）语法分析流程图



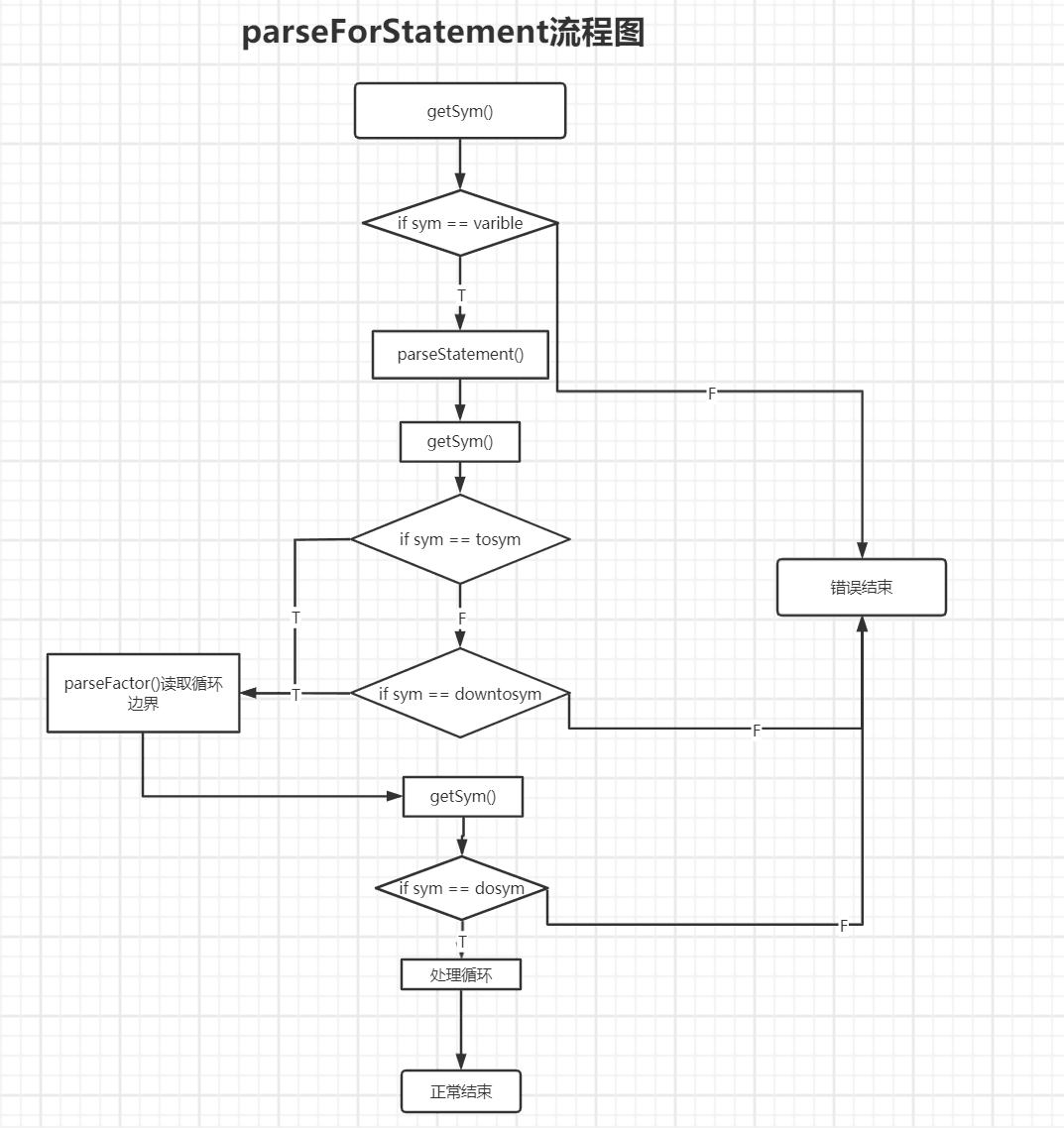
#### （三）IF分析流程图



#### （四）do while流程图

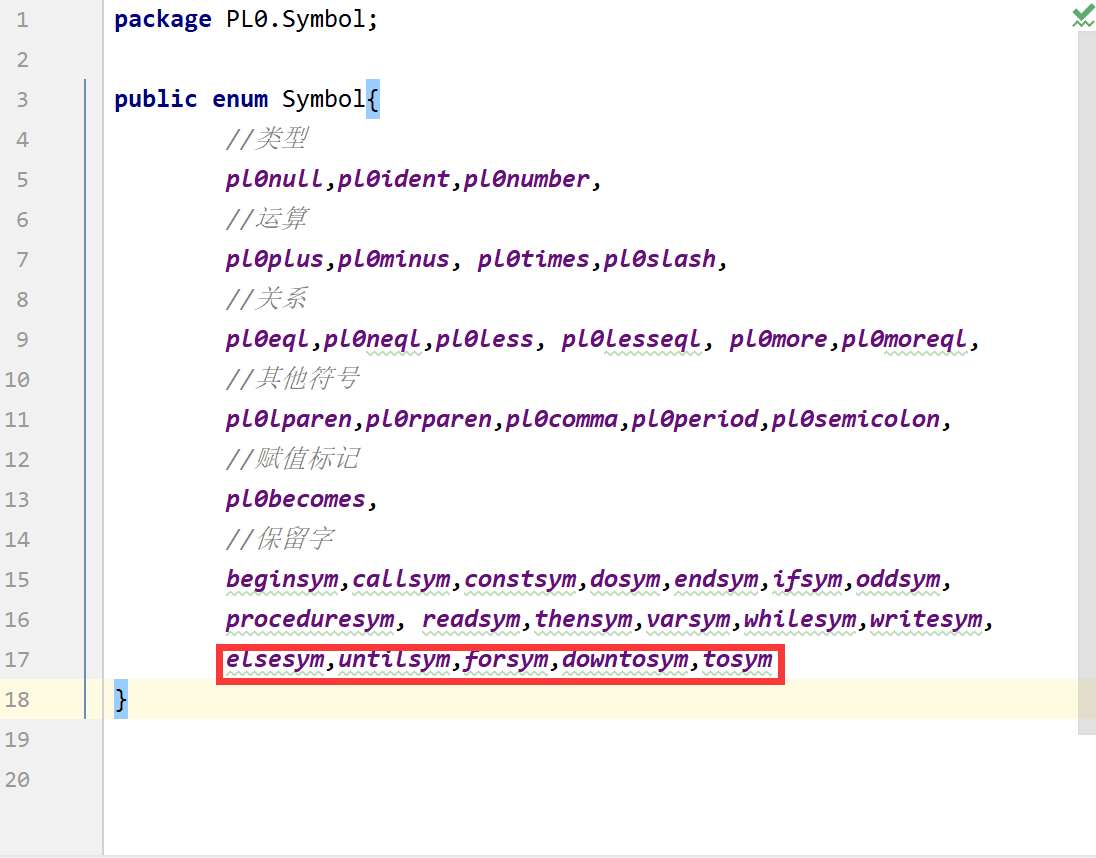


#### （五）For流程图

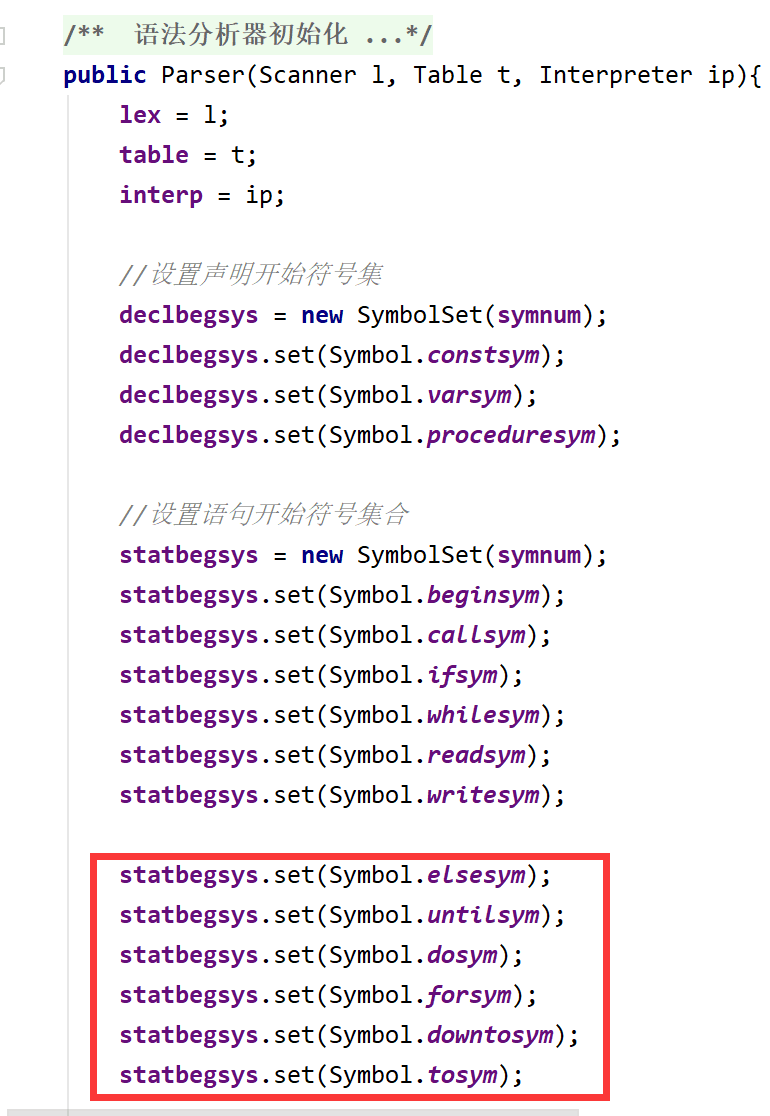


## 四、系统实现

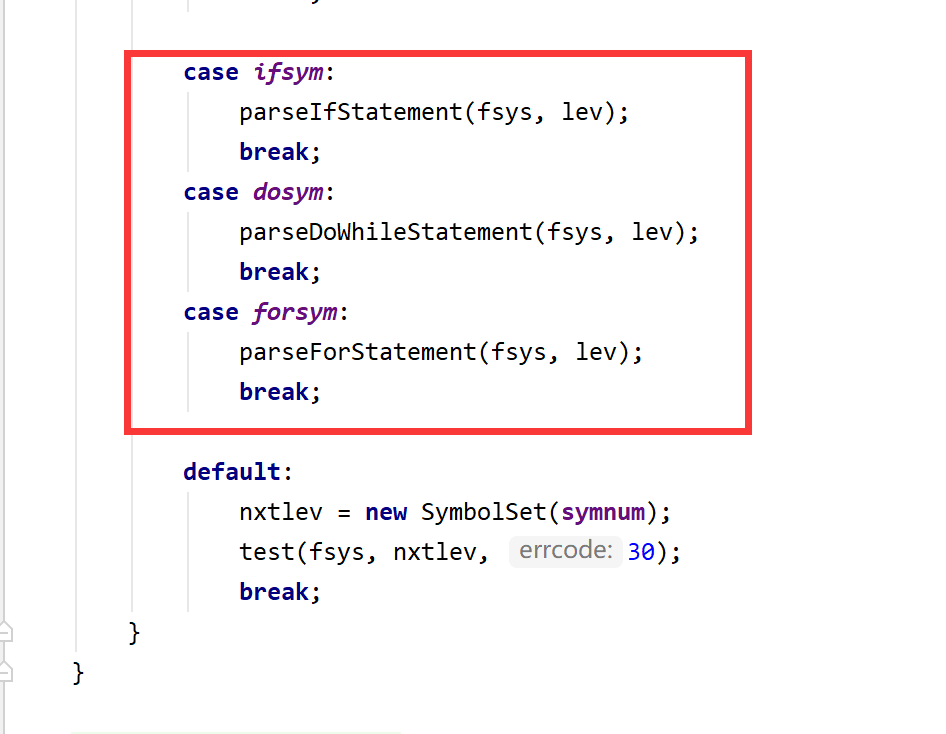
1、在Symbol类中添加新的保留字elsesym、forsym、downtosym、tosym、untilsym。



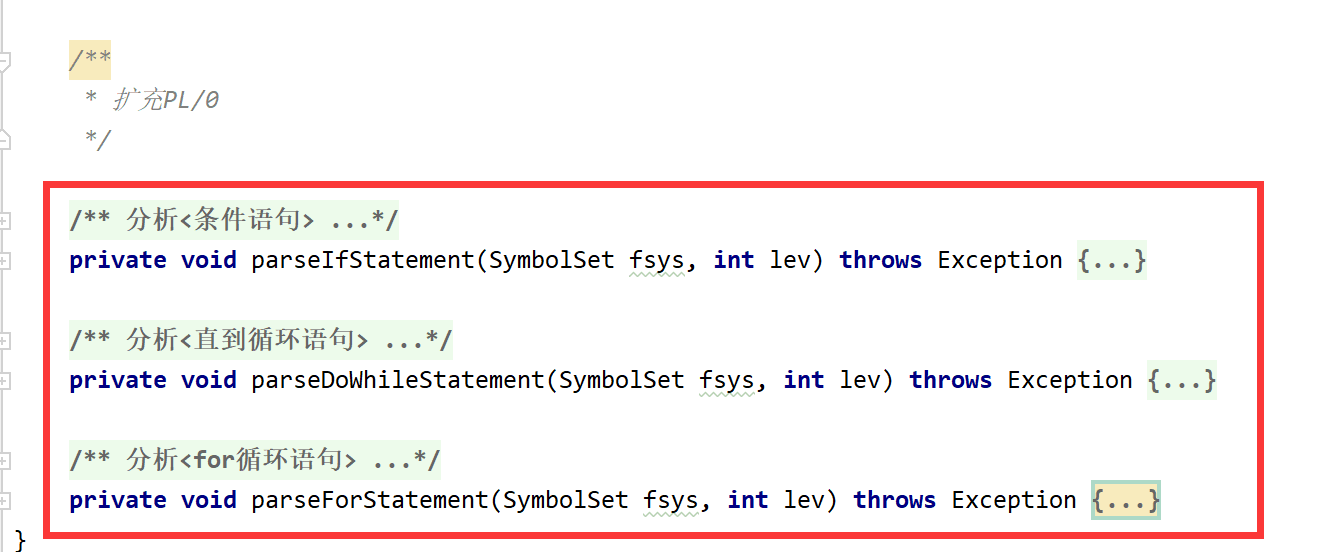
2、在Parser类的构造函数中将上述符号添加到语句开始符号集合中。



3、修改Parser类中的parseStatement()方法。



4、在Parser类中添加parseIfStatement()、parseDoWhileStatement()和parseForStatement()方法。



## 五、程序源代码

#### 1、if else语句

*/\*\*  
 \* 分析<条件语句>  
 \** ***@param*** *fsys 后跟符号集  
 \** ***@param*** *lev 当前层次  
 \*/***private void** parseIfStatement(SymbolSet fsys, **int** lev) **throws** Exception {  
 **int** cx1, cx2;  
 SymbolSet nxtlev;  
  
 nextSym();  
 nxtlev = (SymbolSet) fsys.clone();  
 nxtlev.set(Symbol.thensym);  
 nxtlev.set(Symbol.dosym);  
 parseCondition(nxtlev, lev); *// 分析<条件>* **if** (sym == Symbol.thensym)  
 nextSym();  
 **else** Error.report(15); *// 缺少then* cx1 = interp.index; *// 保存当前指令地址* interp.gen(Fct.JPC, 0, 0); *// 生成条件跳转指令，跳转地址未知，暂时写0* parseStatement(fsys, lev); *// 处理then后的语句* Scanner tmpLex = lex.clone(); *//保存当前Scanner对象，用于在读取下一符号不是else后恢复编译状态* nextSym();  
 nxtlev.set(Symbol.elsesym); *//设置后继符号else* **if**(sym == Symbol.elsesym){  
  
 cx2 = interp.index;  
 interp.gen(Fct.JMP, 0, 0); *// 生产跳转命令，地址未知* nextSym();  
  
 interp.code[cx1].a = interp.index;  
 parseStatement(fsys, lev);  
  
 interp.code[cx2].a = interp.index; *//回填if执行完毕跳转地址* }**else**{  
 lex = tmpLex;  
 sym = lex.sym;  
 interp.code[cx1].a = interp.index; *// 经statement处理后，index为then后语句执行* }  
  
}

#### 2、do while until语句

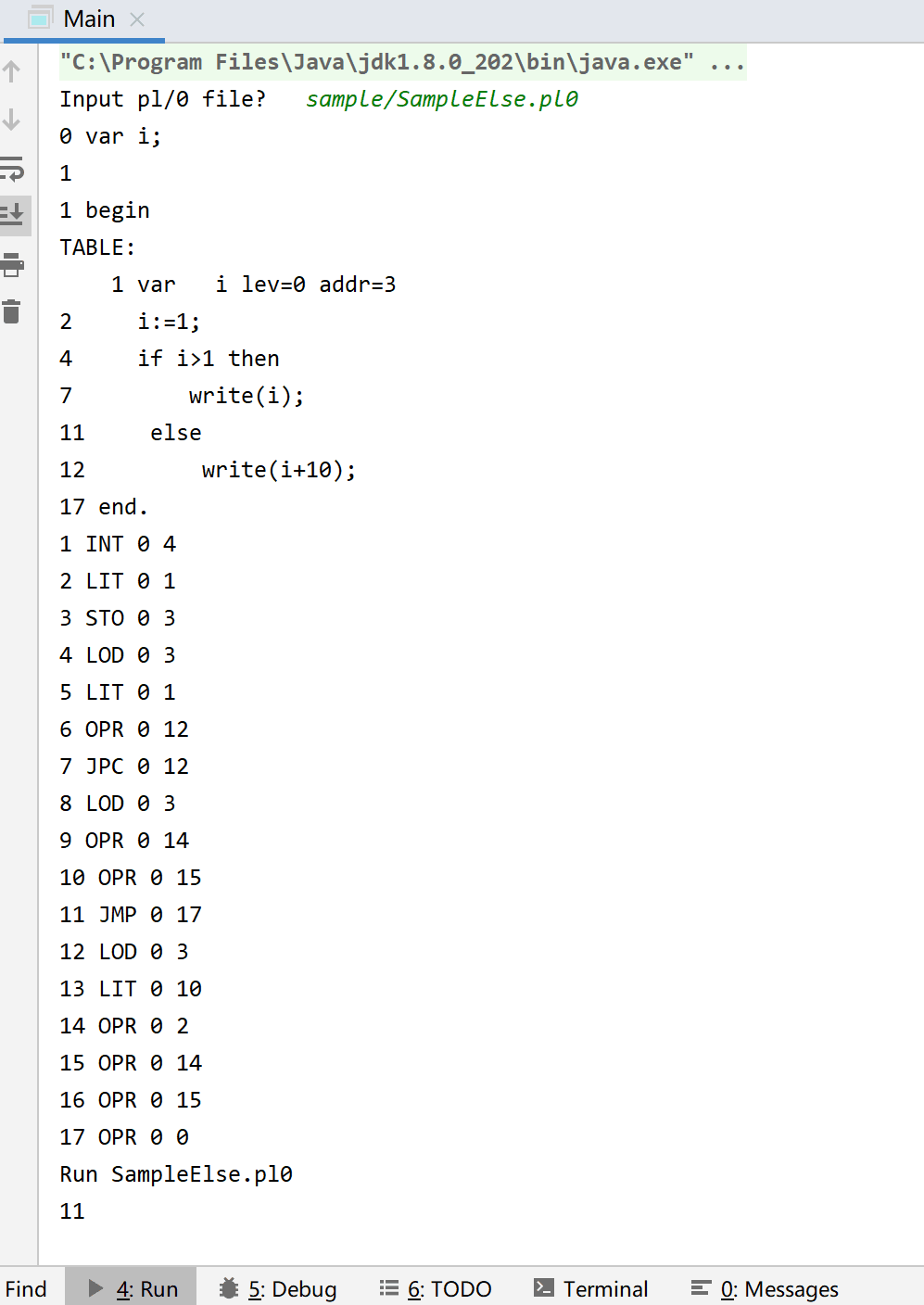
*/\*\*  
 \* 分析<直到循环语句>  
 \** ***@param fsys*** *后跟符号集  
 \** ***@param lev*** *当前层次  
 \*/***private void** parseDoWhileStatement(SymbolSet fsys, **int** lev) **throws** Exception {  
 **int** cx1, cx2;  
 SymbolSet nxtlev;  
  
 nextSym();  
 nxtlev = (SymbolSet) fsys.clone();  
 nxtlev.set(Symbol.***whilesym***); *// 后跟符号为while* **if** (**sym** == Symbol.***whilesym***)  
 nextSym();  
 **else** Error.*report*(18); *// 缺少while* cx1 = **interp**.**index**; *//执行语句块的开始* parseStatement(nxtlev, lev); *//分析<语句>* nextSym();  
 nxtlev.set(Symbol.***untilsym***); *//后跟符号为until* **if** (**sym** == Symbol.***untilsym***)  
 nextSym();  
 **else** Error.*report*(18); *// 缺少until* parseCondition(nxtlev, lev); *//分析<条件>* cx2 = **interp**.**index**; *// 条件判断语句位置* **interp**.gen(Fct.***JPC***, 0, 0); *// 生成条件跳转，但跳出循环的地址未知* **interp**.gen(Fct.***JMP***, 0, cx1); *// 跳转到语句块开始* **interp**.**code**[cx2].**a** = **interp**.**index**; *//回填当前地址*}

#### 3、for语句

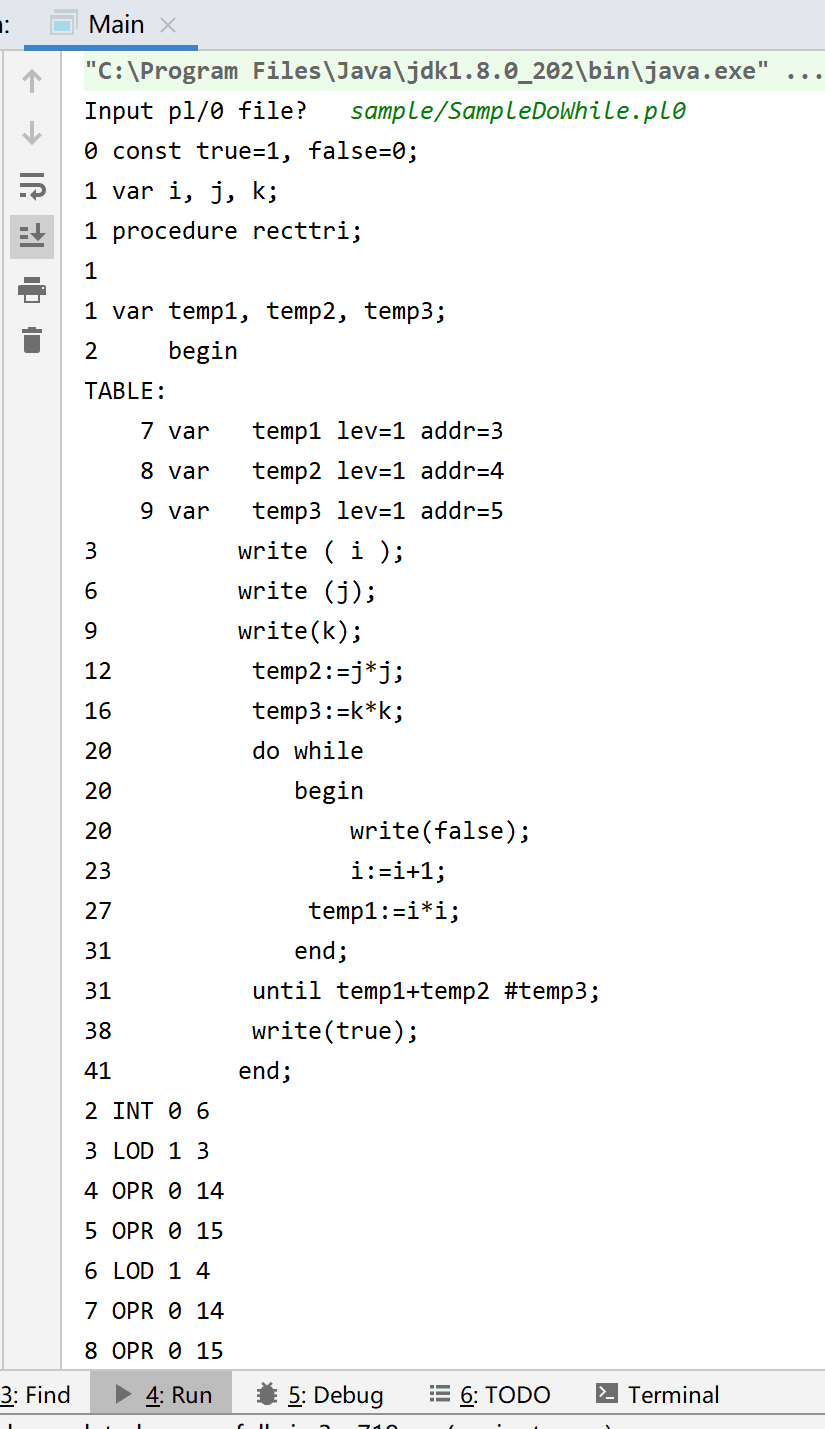
*/\*\*  
 \* 分析<for循环语句>  
 \** ***@param fsys*** *后跟符号集  
 \** ***@param lev*** *当前层次  
 \*/***private void** parseForStatement(SymbolSet fsys, **int** lev) **throws** Exception {  
 **int** cx1 = 0,cx2;  
 SymbolSet nxtlev;  
  
 nextSym();  
 nxtlev = (SymbolSet) fsys.clone();  
  
 **int** i = **table**.position(**lex**.**id**); *//循环控制变量位置* TableItem item = **table**.get(i);  
 **if**(item.**kind** == Pl0ObjectKinds.***variable***) {  
  
 Pl0Variable variable = **table**.getVariable(i);  
  
 parseStatement(fsys,lev);  
  
 nxtlev.set(Symbol.***tosym***);  
 nxtlev.set(Symbol.***downtosym***); *//后跟to、downto* Symbol mode = **sym**;  
  
 cx1 = **interp**.**index**;  
 **interp**.gen(Fct.***LOD***, lev-variable.**level**,variable.**address**); *//循环控制变量读取到栈顶  
  
 //根据不同的模式，有不同的循环判断条件* **if**(**sym** == Symbol.***tosym***){  
 nextSym();  
 parseFactor(fsys,lev); *//读取循环边界（将循环边界读取到栈顶）* **interp**.gen(Fct.***OPR***, 0, 13); *//循环边界是否大于循环控制变量* }**else if**(**sym** == Symbol.***downtosym***){  
 nextSym();  
 parseFactor(fsys,lev); *//读取循环边界（将循环边界读取到栈顶）* **interp**.gen(Fct.***OPR***, 0, 11); *//循环边界是否小于循环控制变量* }**else**{  
 Error.*report*(33); *//for关键字后需要to、downto* }  
 cx2 = **interp**.**index**;  
 **interp**.gen(Fct.***JPC***,0,0); *//条件跳转，不满足条件跳转到循环结束，暂定为0* nxtlev.set(Symbol.***dosym***);  
 **if**(**sym** == Symbol.***dosym***){  
 nextSym();  
 parseStatement(fsys,lev); *//执行语句块（循环体）* }**else**{  
 Error.*report*(34); *//to、downto关键字后需要do* }  
  
 **interp**.gen(Fct.***LOD***, lev-variable.**level**, variable.**address**); *//将循环控制变量读到栈顶  
 /\*\*  
 \* 根据不同模式循环判断变量增大或减小  
 \* 这里为了方便，在Interpreter中定义了新的OPR 17与18。  
 \* 分别用于将栈顶的元素加一和减一  
 \*/* **if**(mode == Symbol.***tosym***){  
 **interp**.gen(Fct.***OPR***,0,17); *//新定义的OPR使得栈顶自增* }**else if**(mode == Symbol.***downtosym***) {  
 **interp**.gen(Fct.***OPR***,0,18); *//新定义的OPR使得栈顶自减* }  
 *//存储循环判断变量* **interp**.gen(Fct.***STO***, lev-variable.**level**, variable.**address**); *//存储循环控制变量  
 //跳转到循环判断语句* **interp**.gen(Fct.***JMP***, 0, cx1); *//跳转到条件判断* **interp**.**code**[cx2].**a** = **interp**.**index**; *//回填条件跳转语句中代码的结尾地址* }**else**{  
 Error.*report*(35); *//for关键字后需要变量赋值表达式* }  
}

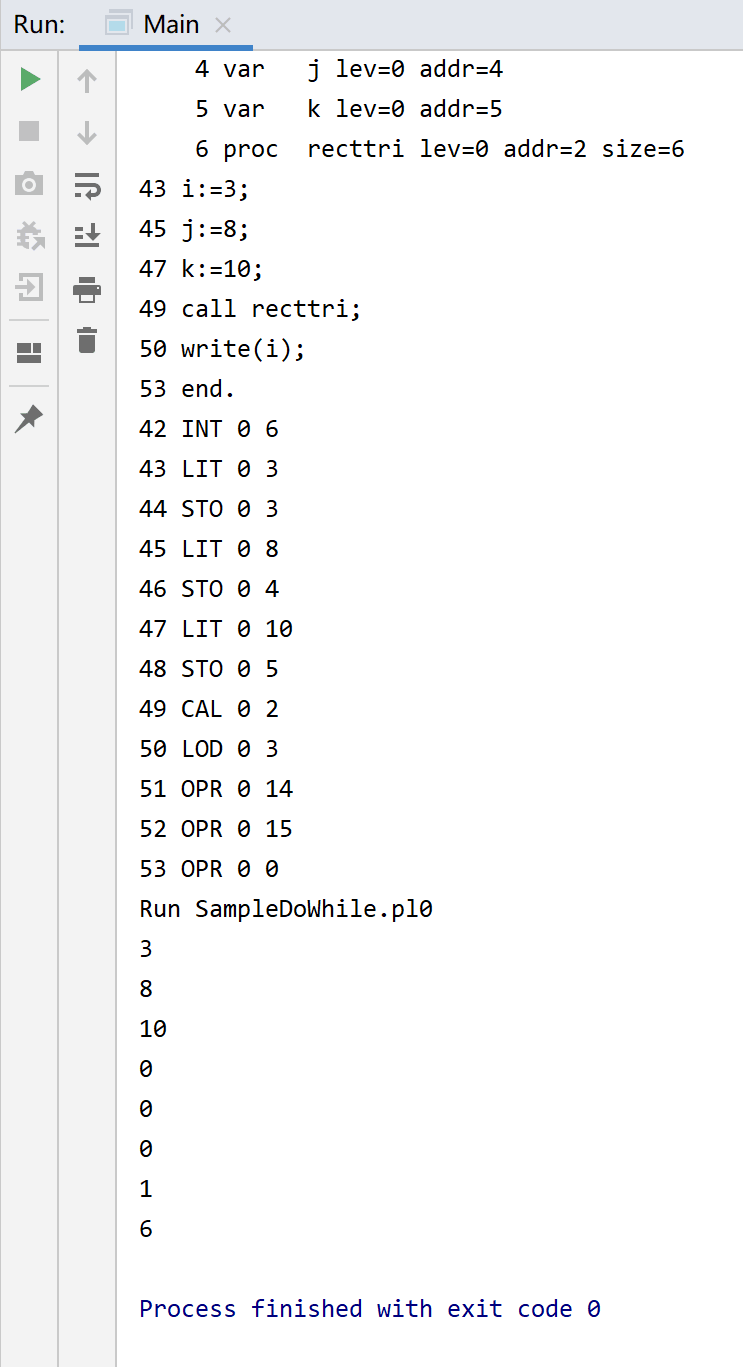
## 六、系统测试

#### 1、if else语句

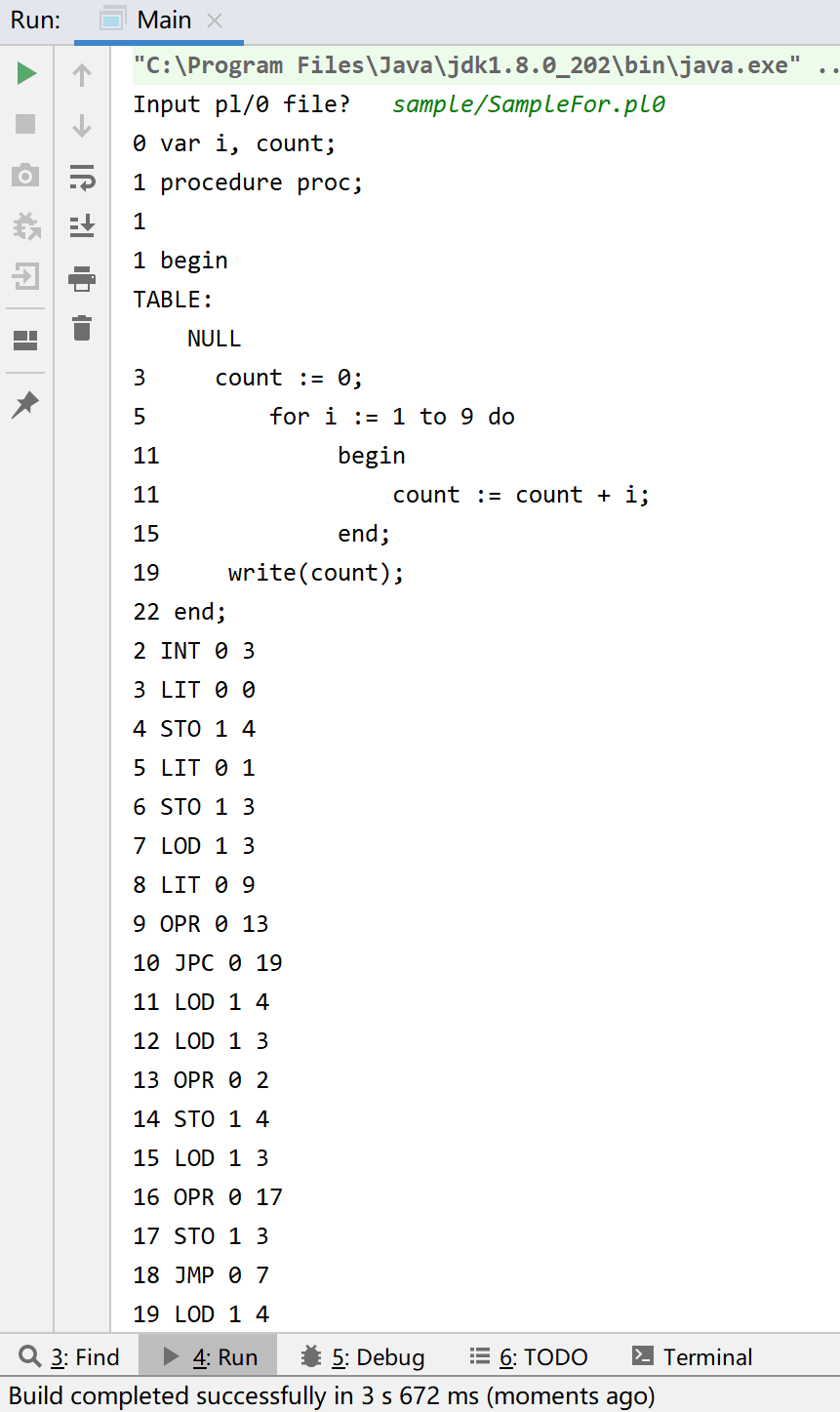


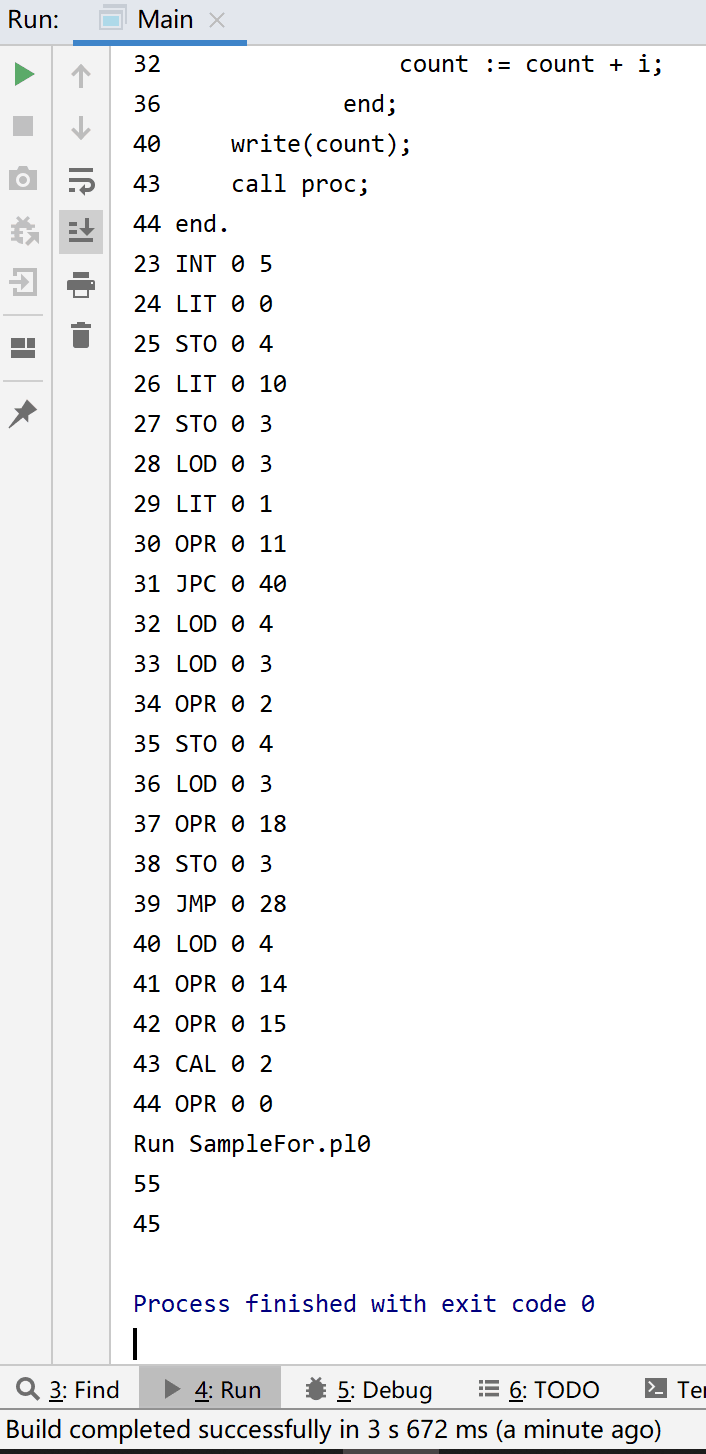
#### 2、do while until语句





#### 3、for语句





## 七、设计总结

实践是学习编译原理和技术的重要环节。对PL/0编译器进行学习和扩充，有助于我们对编译器建立起基本的认识，体验到构造编译器的乐趣。通过本次课程设计，我们小组成员对整个PL/0编译器的结构和过程有了深刻的理解并尝试进行了扩充，这对我们的编程能力有很大的提高，并能在编程过程中思考计算机底层的工作流程。

## 八、课题成员及分工

## 九、自评成绩