

编译技术课程实验报告

|  |  |
| --- | --- |
| 实验名称： | 实验二  语法制导的三地址代码生成程序 |
| 实验日期： | 2024-4-30 |
| 实验地点： | 西部片区4号楼208 |
|  | |
| 学号： | 33920212204567 |
| 姓名： | 任宇 |
| 专业年级： | 软工2021级 |
| 学年学期： | 2023-2024学年第二学期 |

目录

[一、 实验目的 3](#_Toc164149851)

[二、 实验内容 3](#_Toc164149852)

[三、 实验环境 3](#_Toc164149853)

[四、 实验过程 3](#_Toc164149854)

[1) 语法制导定义 3](#_Toc164149855)

[2) 改写后的产生式集合 4](#_Toc164149856)

[3) 化简后的语法图 4](#_Toc164149857)

[4) 递归子程序的算法 4](#_Toc164149858)

[5) 三地址代码生成器的数据结构 7](#_Toc164149859)

[6) 程序结构的说明 9](#_Toc164149860)

[7) 运行结果 11](#_Toc164149861)

[五、 思考题 11](#_Toc164149862)

[六、 实验心得 11](#_Toc164149863)

# 实验目的

掌握计算机语言的语法分析程序设计与属性文法应用的实现方法。

# 实验内容

编制一个能够进行语法分析并生成三地址代码的微型编译程序。

# 实验环境

* PC微机 Windows10操作系统
* 开发环境为VS2022

# 实验过程

## 语法制导定义

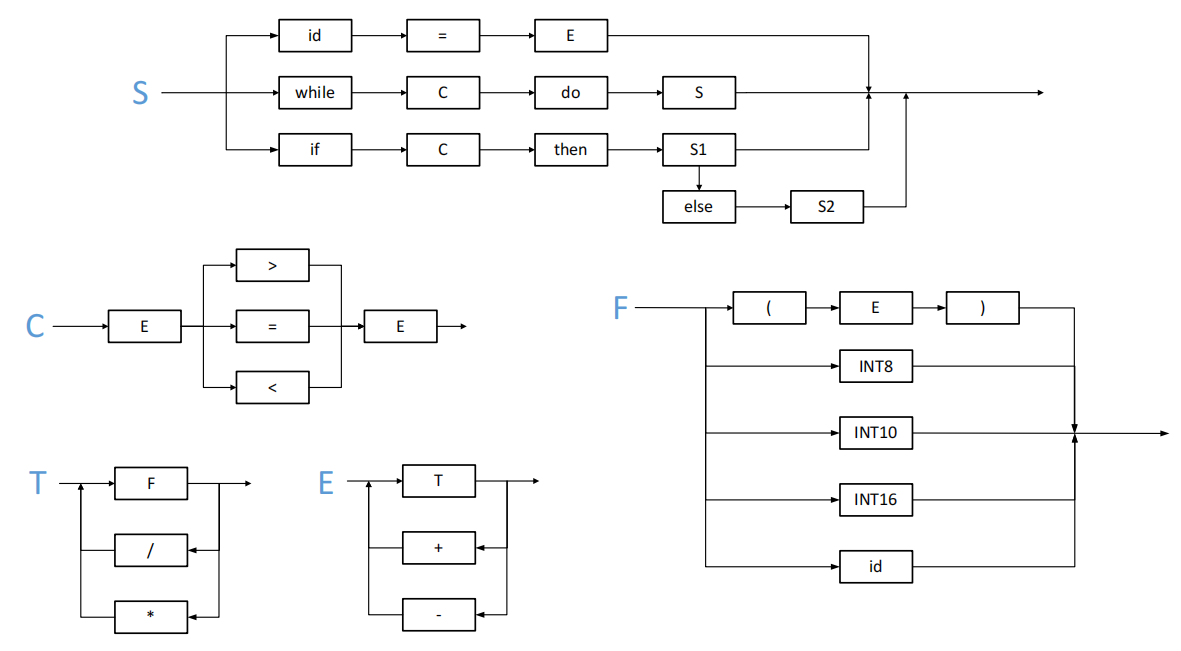
|  |  |
| --- | --- |
| 产生式 | 语义规则 |
| S → id = E; | S.code = E.code || gen(id.place ‘:=’ E.place) |
| S → if C then S1; | C.true = newlabel; C.false = S.next; S1.next = S.next; S.code = C.code || gen(C.true’:’) || S1 .code |
| S → if C then S1 else S2; | C.true = newlabel; C.false = newlabel;  S1.next = S2.next = S.next;  S.code = C.code || gen(C.true ‘:’) || S1 .code || gen(‘goto’ S.next) || gen(C.false ‘:’) || S2.code; |
| S → while C do S1; | S.begin = newlabel; C.true = newlabel;  C.false = S.next; S1.next = S.begin;  S.code = gen(S.begin ‘:’) || C.code || gen(C.true ‘:’) || S1.code || gen(‘goto’ S.begin); |
| C → E1 > E2 | C.code = E1.code || E2.code || gen(‘if’ E1.place ‘>’ E2.place ‘goto’ C.true) || gen(‘goto’ C.false) |
| C → E1 < E2 | C.code = E1.code || E2.code ||gen(‘if’ E1.place ‘<’ E2.place ‘goto’ C.true) || gen(‘goto’ C.false) |
| C → E1 = E2 | C.code = E1.code || E2.code ||gen(‘if’ E1 .place ‘=’ E2.place ‘goto’ C.true) || gen(‘goto’ C.false) |
| E → E1 + T | E.place = newtemp;  E.code = E1.code || T.code || gen(E.place ‘:=’ E1 .place ‘+’ T.place) |
| E → E1 - T | E.place = newtemp;  E.code = E1.code || T.code || gen(E.place ‘:=’ E1.place ‘-’ T.place) |
| E → T | E.place = T.place; E.code = T.code |
| T → F | T.place = F.place; T.code = F.code |
| T → T1 \* F | T.place = newtemp;  T.code = T1.code || F.code || gen(T.place ‘:=’ T1.place ‘\*’ F.place) |
| T → T1 / F | T.place = newtemp;  T.code = T1.code || F.code || gen(T.place ‘:=’ T1.place ‘/’ F.place) |
| F → (E) | F.place = E.place; F.code = E.code |
| F → id | F.place = id.name; F.code = ‘’ |
| F → INT8 | F.place = int8.value; F.code = ‘’ |
| F → INT10 | F.place = int10.value; F.code = ‘’ |
| F → INT16 | F.place = int16.value; F.code = ‘’ |

## 改写后的产生式集合

消除左递归和提取左因子，如下表所示：

|  |  |
| --- | --- |
| 原产生式 | 修改后的产生式 |
| C → E1 > E2  C → E1 < E2  C → E1 = E2 | C → E’  E’ → >E | =E | <E |
| E → E1 + T  E → E1 - T  E → T | E → TE’  E’ → + TE’ | - TE’ |
| T → F  T → T1 \* F  T → T1 / F | T → FT’  T’ → \* FT’ | / FT’ |

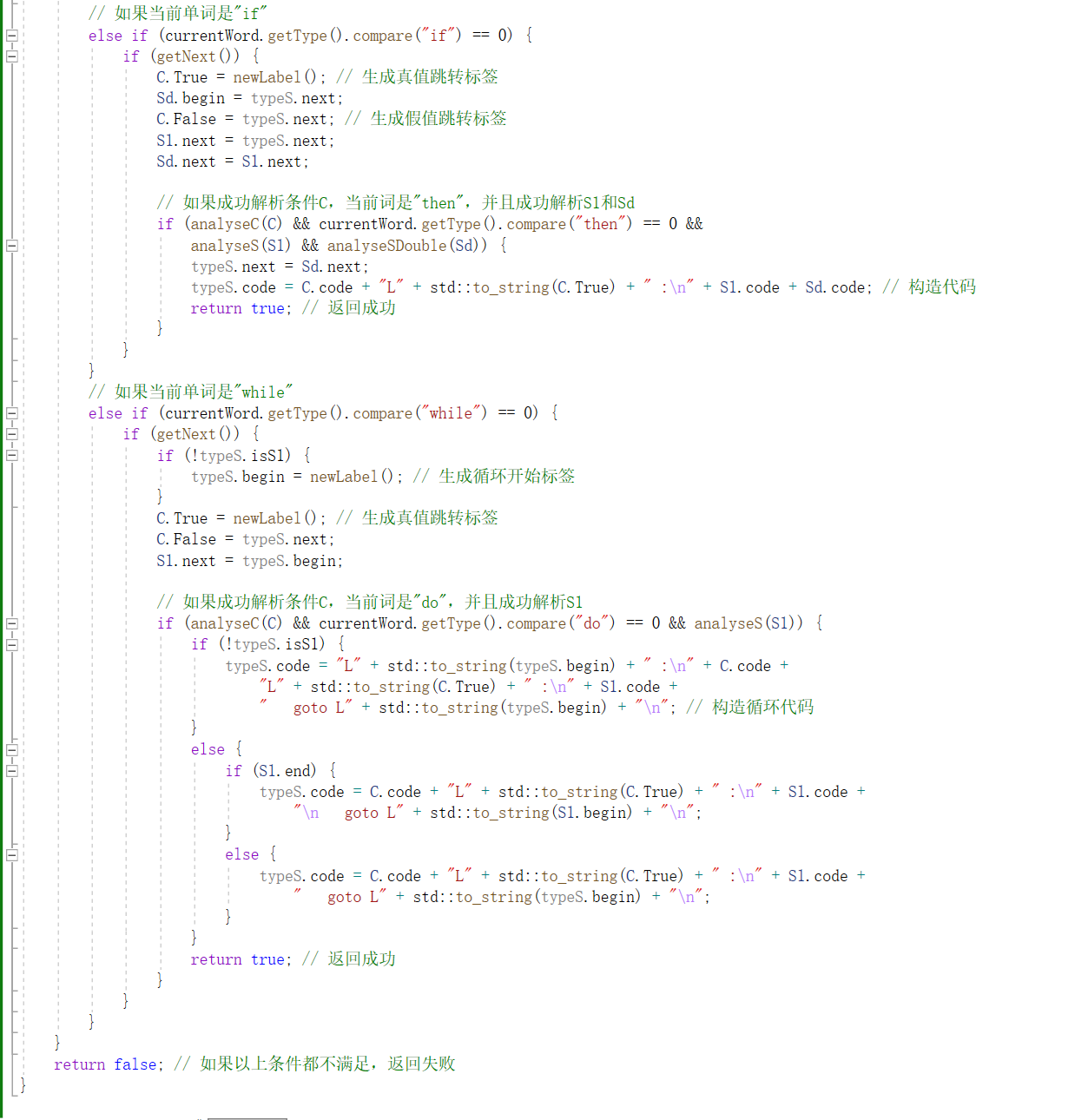
## 化简后的语法图



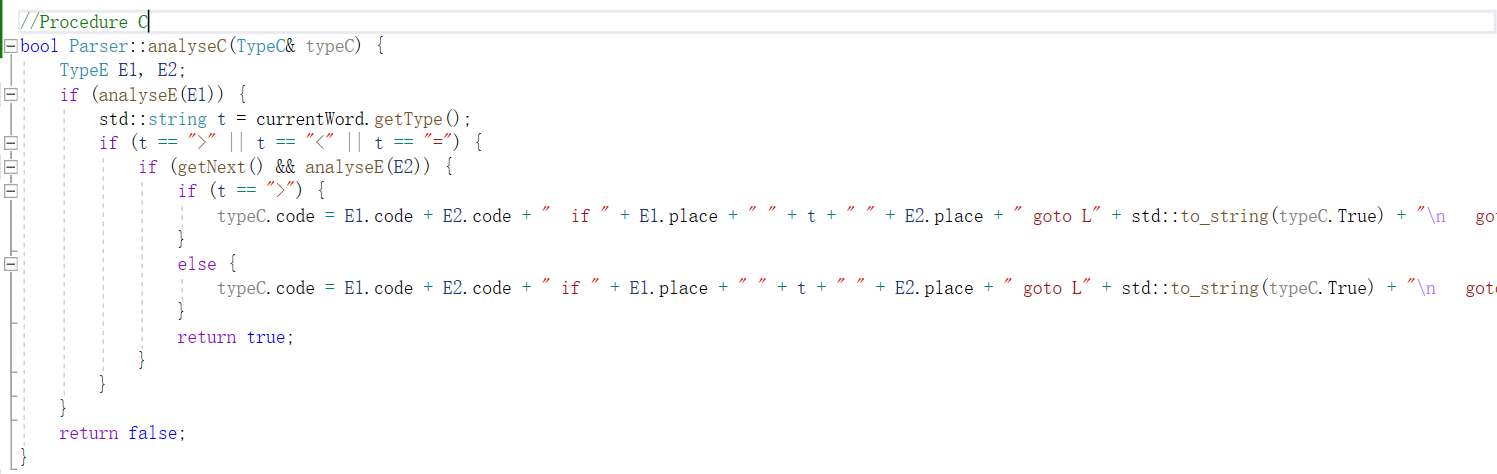
## 递归子程序的算法

Procedure S





Procedure C



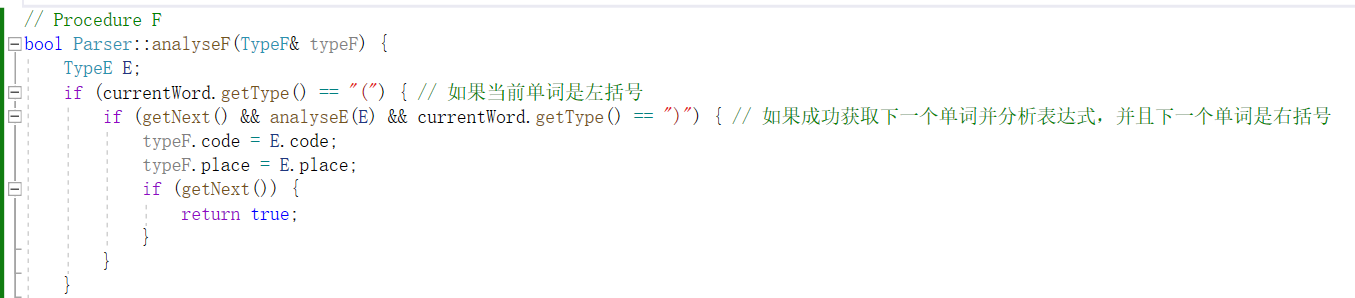
Procedure E



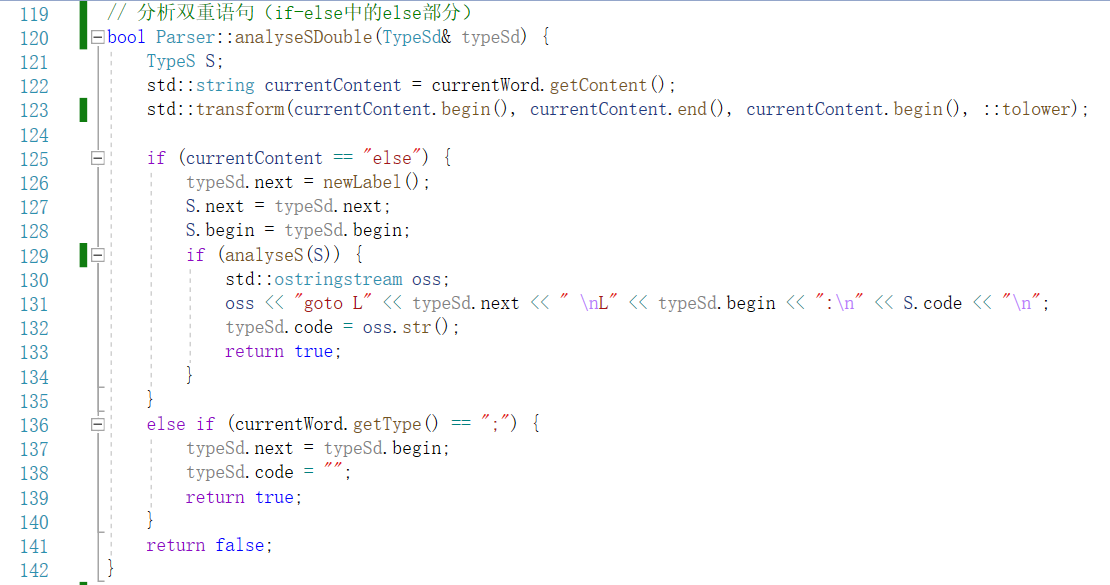
Procedure T



Procedure F

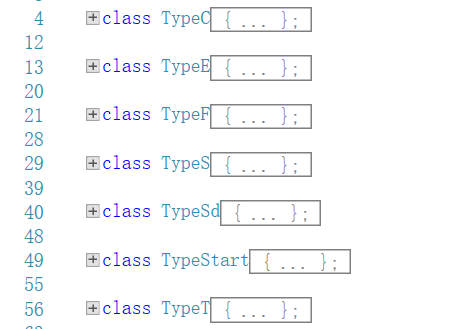


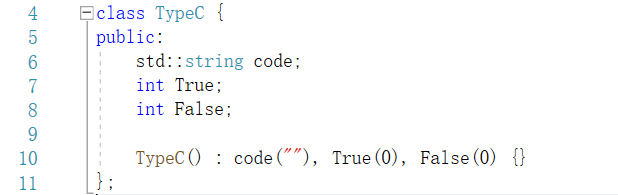
Procedure SD(用于处理if-else)

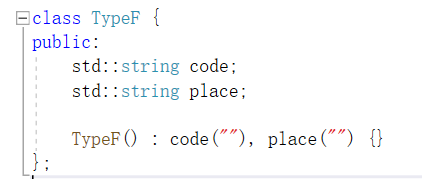
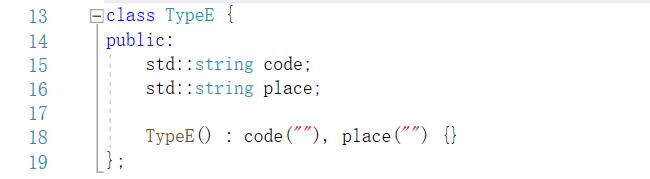


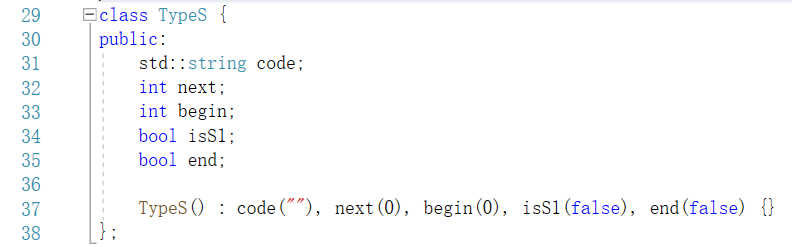
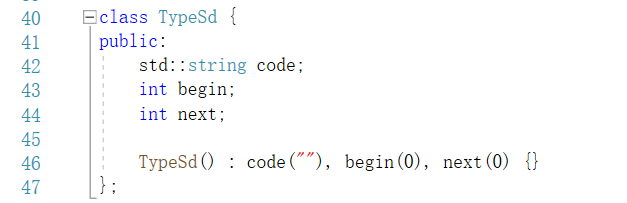
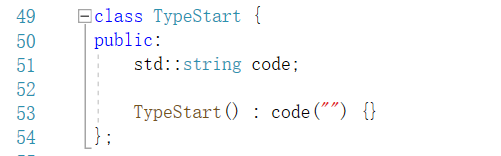
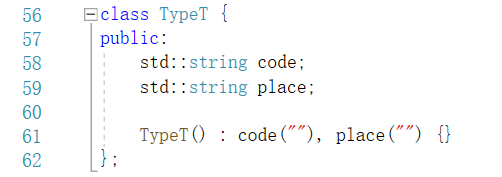
## 三地址代码生成器的数据结构

非终结符的中间代码结构体：







## 程序结构的说明

* Word 类

属性：

* content：单词的实际字符串值。
* type：单词的分类（如标识符、关键字、整数等）。

方法：

初始化构造函数。

* + setContent、setType：设置单词的内容和类型。
  + getContent、getType：获取单词的内容和类型。
* WordScan 类、

属性：

* keyWords：预定义的关键词列表。
* operators：表示操作符的字符列表。

方法：

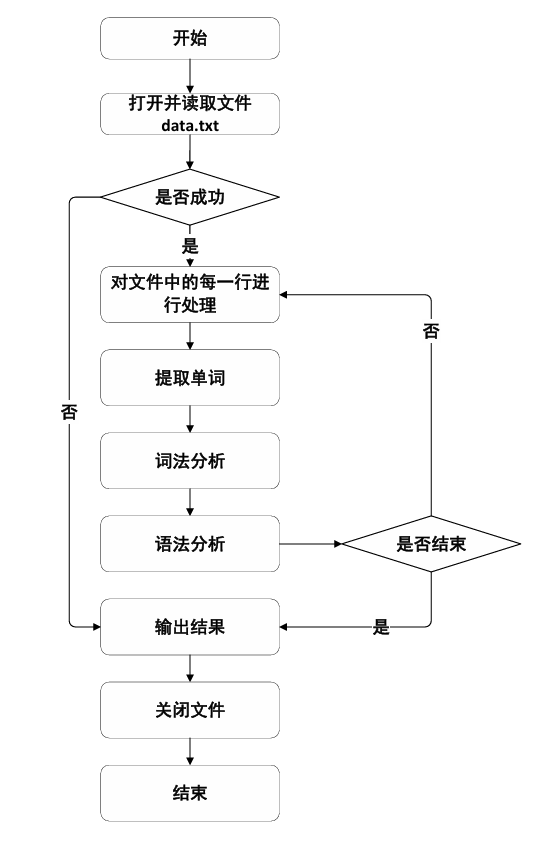
* scan：根据特定标准分析字符串，识别并分类所有组成部分（关键词、标识符、数字、操作符），并将它们添加到结果向量中。
* 处理数字的方法（decimalNumbers、octalNumbers、hexadecimalNumbers）：提取并分类不同进制的数字。
* idn：识别标识符。
* digit：根据数字系统处理数字的小数部分。
* isOtherString：分类未被预定义类型识别的字符。
* isKeyWords：检查字符串是否为关键词。
* Parser 类

属性：

* words：待解析的Word对象列表。
* currentWord：当前正在分析的词。
* label、val：在生成代码中用于生成标签和临时变量的计数器。

方法：

* + getNext：移动到列表中的下一个词。
  + newTemp：生成一个新的临时变量。
  + setWords：用单词列表初始化解析器。
  + 解析方法（Start、analyseS、analyseSDouble、analyseC、analyseE、analyseT、analyseF）：每个方法根据简单的语法处理特定部分，生成中间代码作为输出。
* 主函数
  + 从文件data.txt读取输入。
  + 对文件中的每一行进行处理，提取单词，然后通过WordScan进行词法分析和通过Parser进行语法分析。
  + 根据语法分析的结果输出生成的中间代码或错误信息。
* 程序的执行流程如下：

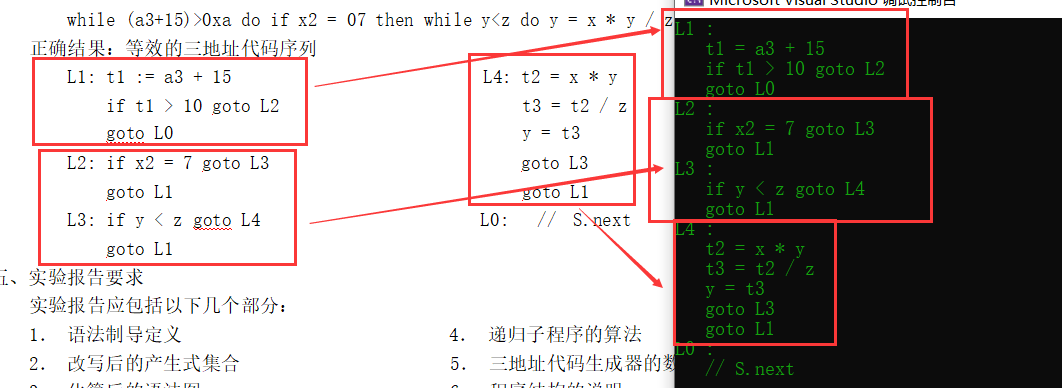


## 运行结果

测试用例为：

while (a3+15)>0xa do if x2=7 then while y<z do y = x \* y / z ;

测试结果为：



# 思考题

1. 生成的三地址代码可否直接输出（不采用数据结构来实现属性code）？

答：不可以，因为三地址代码的生成是依靠文法符号子过程递归调用实现的，如果三地址码直接输出，不采用数据结构来实现属性code，可能会使得输出混乱无序。同时，上一层的代码输出依赖于调用的子过程代码输出，不使用数据结构来存储对应的值，上层代码无法输出。

1. 如何保证四则运算的优先关系和左结合性？

答：三地址代码的实现依赖于属性文法，而属性文法局限于文法产生式，因此只要将文法规范化，消除左递归，提取左因子，产生的代码就会符合四则运算的优先关系和左结合性。

1. 如何采用代码段相对地址代替三地址代码序列中的标号？

答：可以在表示非终结符的中间代码结构体中加入地址属性，表示该变量对应的三地址码的起始地址。

# 实验心得

通过这次实验，我深入理解了编译器中词法分析和语法分析的核心机制，特别是如何从源代码中提取单词（Token）并构建出有效的语法结构来生成三地址代码。实验过程中，使用属性文法来定义语义动作，使我更加明确了编译过程中各个组成部分之间的联系和作用。此外，本次实验也加深了我对理论知识的应用理解，为以后深入学习编译原理打下了坚实的基础。