目录

[一、工作目标及介绍 2](#_Toc60084492)

[二、设计思路 3](#_Toc60084493)

[2.1编译程序概述 3](#_Toc60084494)

[2.2总体设计 4](#_Toc60084495)

[2.3词法分析 5](#_Toc60084496)

[2.3.1词法分析设计思路 7](#_Toc60084497)

[2.3.2关键词和标识符 7](#_Toc60084497)

[2.3.3相关词法规则 7](#_Toc60084497)

[2.3.4等等 7](#_Toc60084497)

[2.4语法分析 7](#_Toc60084497)

[2.4.1语法分析设计思路 7](#_Toc60084497)

[2.4.2文法产生式以及DFA状态 7](#_Toc60084497)

[2.4.3 SLR(1)分析表 7](#_Toc60084497)

[2.4.4等等 7](#_Toc60084497)

[2.5 语义分析 12](#_Toc60084499)

[2.4.1语义分析设计思路 7](#_Toc60084497)1

[2.4.2四元式中间代码 7](#_Toc60084497)

[2.4.3归约动作相应的语义规则 7](#_Toc60084497)

[2.4.4等等 7](#_Toc60084497)

[三、核心代码 13](#_Toc60084500)

[3.1文件读取(伪代码) 13](#_Toc60084501)

[3.2词法分析(伪代码) 14](#_Toc60084502)

[3.3语法分析（伪代码） 17](#_Toc60084503)

[3.4语义分析（伪代码） 17](#_Toc60084503)

四、[用例测试与结果分析………………………………………………………17](#_Toc60084507)

[4.1测试用例1 18](#_Toc60084508)

[4.2测试用例2 18](#_Toc60084509)

[4.3测试用例3 19](#_Toc60084510)

[4.4测试用例4 2](#_Toc60084510)0

[五、遇到问题及解决 19](#_Toc60084511)

[5.1等等 2](#_Toc60084512)1

[5.2等等 2](#_Toc60084513)1

[5.3等等 21](#_Toc60084514)

[六、心得体会 22](#_Toc60084515)

2.4语法分析

语法分析就算的主要功能就是对从词法分析得到的对输入串转换之后的数字序列按照已经构造好的SLR(1)分析表进行相应的分析，判断输入语句是否合法，同时生成相应的语法树，为语义分析做准备。在本报告中，语义分析与语法分析是同步进行的，所以并未画出具体的语法树。

语法分析的主要原理如下：先设置相应的状态栈、符号栈、输入串，分别用于存储当前的状态序列、符号序列以及输入序列，输入序列会伴随着移进操作逐步进入符号栈，符号栈中的符号会在规约操作时出栈，并将相应的非终结符压入栈。状态栈中的数字序列会伴随着移进、规约操作而与输入序列、符号栈同步变化。

一开始分别将0、#压入状态栈和符号栈，之后读入输入串的首位元素，将状态栈的栈顶元素作为索引的第一部分key1，输入串的首位元素作为索引的第二部分key2，进入SLR(1)分析表中进行查找，如果根据两个索引可以找到相应的操作，则根据索引得到的结果action中的第一部分flag判断是移进还是规约操作。

如果是规约操作，先将状态栈和符号栈都做相应产生式右侧字符数量的出栈操作，之后将新的状态栈的栈顶元素作为索引的第一部分key1，规约之后的非终结符作为索引的额第二部分key2，继续根据SLR(1)分析表进行操作，但此时仅需要对状态栈进行操作，符号栈没有操作。

重复上述步骤直至输入字符串为空同时状态栈仅剩余01，符号栈仅剩余S即可判定输入字符串符合语法，输出acc。

2.4.1语法分析设计思路

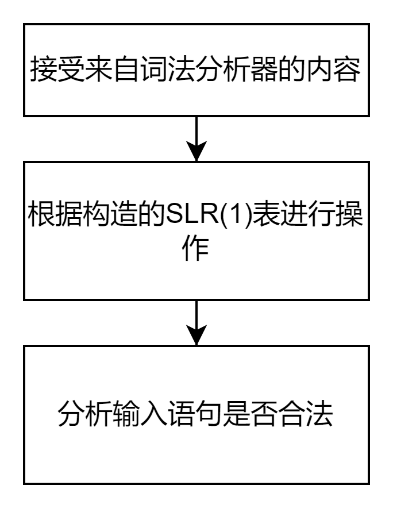


图2-4-1语法分析流程图

2.4.2文法产生式以及DFA状态

0 S’->S

1 S->id := E

2 S->if E then M S

3 S->if E then M S N else M S

4 S->while M E do M S

5 S->S; M S

6 M->ε

7 N->ε

8 E->E + E

9 E->E \* E

10 E->- E

11 E->id

12 E->E || M E

13 E->E && M E

14 E->not E

15 E->(E)

16 E->E rop E

17 E->True

18 E->False

其中：S为语句，M、N为语义输出时必须符号，E为复合控制语句

2.4.3 DFA

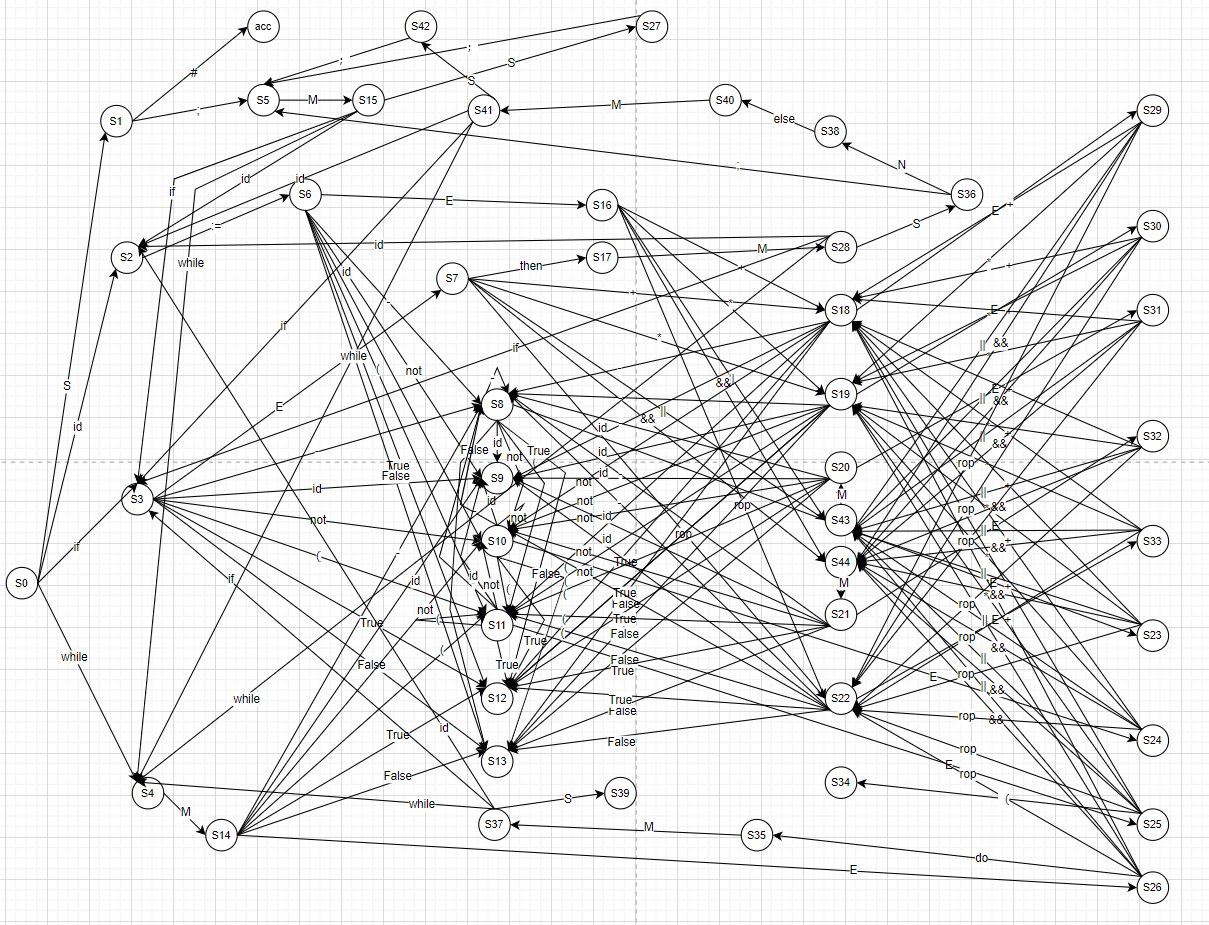


图2-4-2 DFA

其中：

S0:

S’->.S

S->.id:=E

S->.if E then M S

S->.if E then M S N else M S

S->.while M E do M S

S->.S;M S

S1:

S’->S.

S->S.;M S

S2:

S->id.:=E

S3:

S->if. E then M S

S->if. E then M S N else M S

E->.E + E

E->.E \* E

E-> .- E

E->.id

E->.E || M E

E->.E && E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S4:

S->while. M E do M S

M->.ε

M->ε.

S5:

S->S;. M S

M->.ε

M->ε.

SS6:

S->id :=.E

E->.E + E

E->.E \* E

E->.- E

E->.id

E->.E | | M E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S7:

S->if E .then M S

S->if E .then M S N else M S

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && E

E->E. rop E

S8:

E->-.E

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E | | M E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S9:

E->id.

S10:

E->not .E

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E | | M E

E->.E && E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S11:

E->(.E)

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E || M E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S12:

E->True.

S13:

E->False.

S14:

S->while M. E do M S

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E | | M E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S15:

S->S; M .S

S->.id := E

S->.if E then M S

S->.if E then M S N else M S

S->.while M E do M S

S16:

S->id := E.

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S17:

S->if E then .M S

S->if E then .M S N else M S

M->.ε

M->ε.

S18:

E->E + .E

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E | | M E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S19:

E->E \* .E

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E | | M E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S20:

E->E | | M .E

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E | | M E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S21:

E->E && M .E

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E || E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S22:

E->E rop .E

E->.E + E

E->.E \* E

E->.-E

E->.id

E->.E | | M E

E->.E && M E

E->.not E

E->.(E)

E->.E rop E

E->.True

E->.False

S23:

E->-E.

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S24:

E->not E.

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S25:

E->(E.)

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S26:

S->while M E. do M S

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S27:

S->S; M S.

S->S.; M S

S28:

S->if E then M .S

S->if E then M .S N else M S

S->.id := E

S->.if E then M S

S->.if E then M S N else M S

S->.while M E do M S

S->.S; M S

S29：

E->E + E.

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S30:

E->E \* E.

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S31:

E->E | | M E.

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S32:

E->E && M E.

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S33:

E->E rop E.

E->E. + E

E->E. \* E

E->E. | | M E

E->E. && M E

E->E. rop E

S34:

E->(E).

S35:

S->while M E do .M S

M->.ε

M->ε.

S36:

S->if E then M S.

S->if E then M S. N else M S

S->S.; M S

N->.ε

N->ε.

S37:

S->while M E do M .S

S->.id := E

S->.if E then M S

S->.if E then M S N else M S

S->.while M E do M S

S->.S; M S

S38:

S->if E then M S N .else M S

S39:

S->while M E do M S.

S->S.; M S

S40：

S->if E then M S N else .M S

M->.ε

M->ε.

S41：

S->if E then M S N else M .S

S->.id := E

S->.if E then M S

S->.if E then M S N else M S

S->.while M E do M S

S->.S; M S

S42：

S->if E then M S N else M S

S->S; M S

S43：

E->E | | .M E

M->.ε

N->ε.

S44:

E->E && .M E

M->.ε

N->ε.

2.4.3 SLR(1)分析表

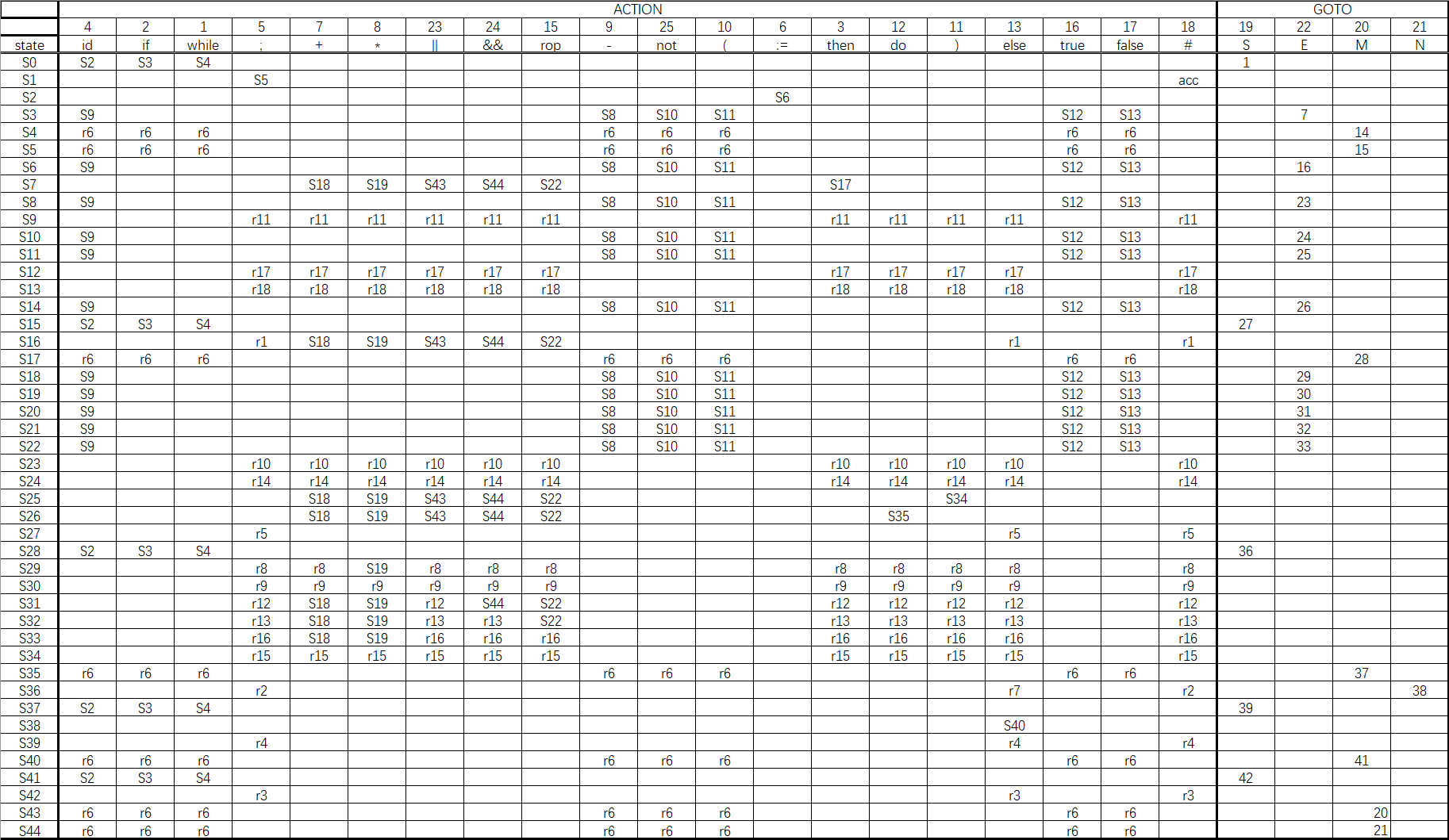


图2-4-3 SLR(1)分析表

2.4.4 语法分析需要的数据

对于语法分析而言，SLR(1)分析表是不可或缺的一部分，在本次课程设计中，采用了map<pair<key1,key2>,action>的数据结构来表示SLR(1)分析表，其中action为结构体含有flag标记操作类型以及，val对应操作的第几条。

同时采用了一个二维数组grammar来记录文法，由于语法分析仅需要知道规约后形成了什么非终结符以及需要规约几个字符，所以grammar中进存储了这两个信息，其中grammar[i][0]是相应产生式左侧的非终结符，grammar[i][1]是产生式右侧的字符数量

3.3 伪代码

语法分析器的操作只有移进和规约，所以对应的数据结构操作只有进出栈，在根据key1和key2查找到相应的动作后进行相应的操作即可。

void SyntaxAnalyzer(){

while(输入串不为空){

输出符号栈、状态栈、输入串

取输入串的第一个元素作为key2;

取状态栈栈顶元素作为key1;

根据(key1,key2)在SLR(1)分析表中进行查询;

if(查询结果不为空)

if(移进操作){

移进

更新状态栈、符号栈

打印移进操作

if(符合结束条件)

打印acc

}

else if(规约操作){

根据key2查找grammer数组，确定规约几位

规约成什么非终结符

更新状态栈、符号栈

打印规约操作、GOTO操作

}

else{

打印错误信息

}

}

}

4 测试用例及结果分析

4.1 测试用例1

if True then a:=c else a:=b

#

测试结果：

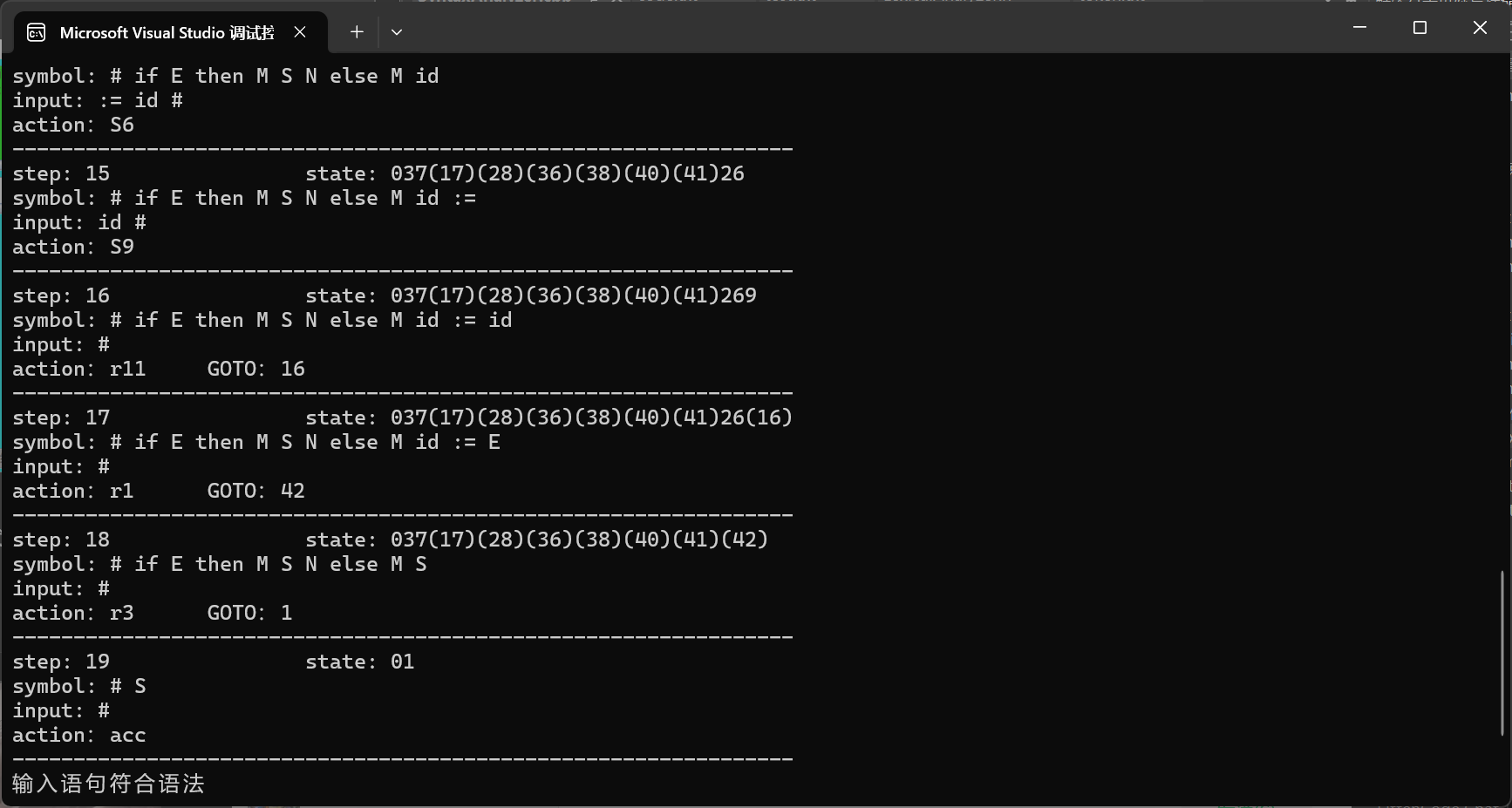


图4-1测试用例1显示输入语句符合语法

4.2 测试用例2（测试用例1去掉了else）

if True then a:=c a:=b

#

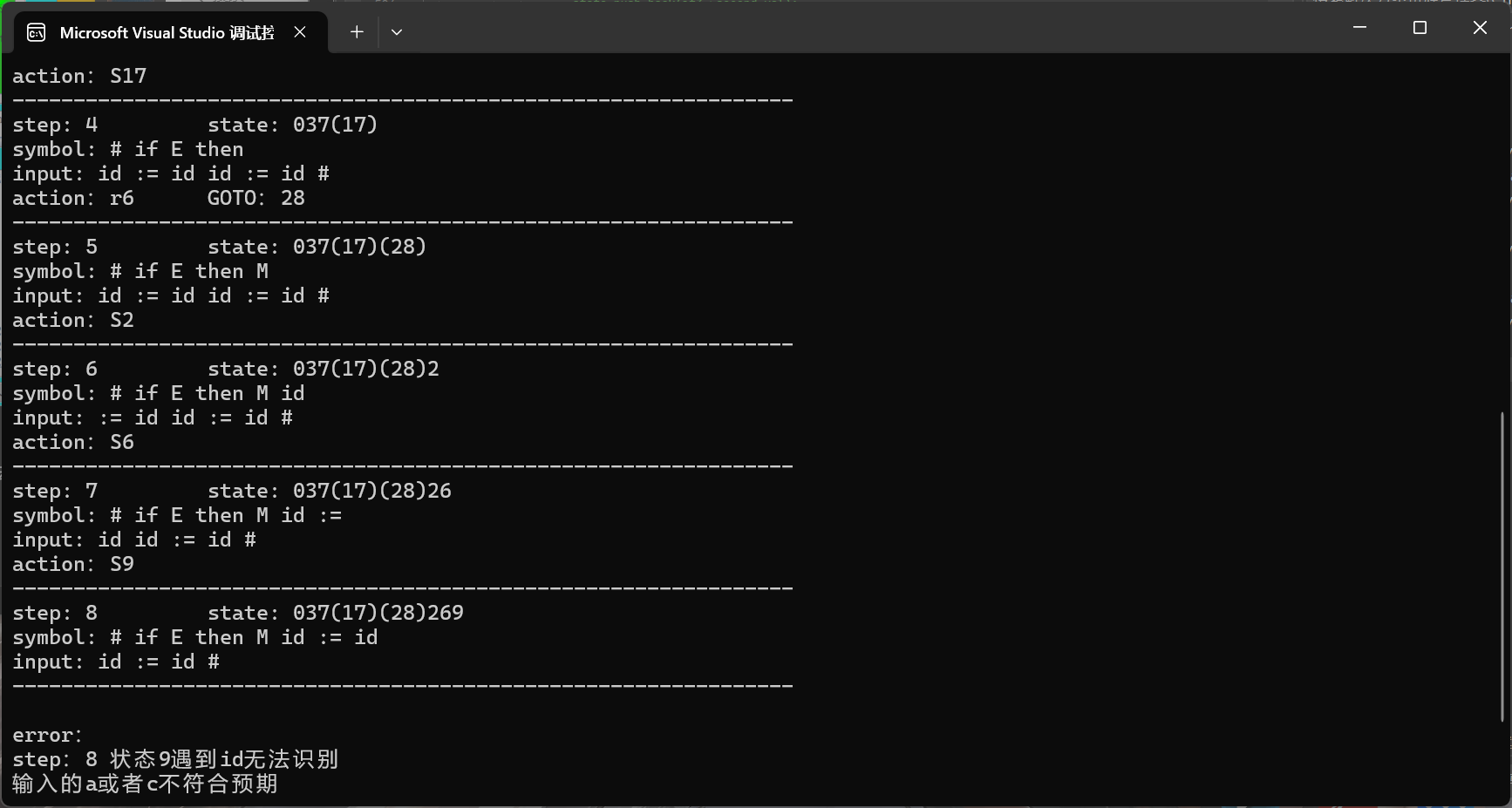


图4-2测试用例2-显示错误信息

从输出结果中发现输入有错误，在原本应该时else的区域输入了不合时宜的a，导致后序步骤无法进行。

4.3 测试用例3

if True || False then a:=(b+c)\*a else a:=-b+c;

while b<=c && a<>c do b:=b+a;

ans:=a+b+c;ans:=ans+d

#

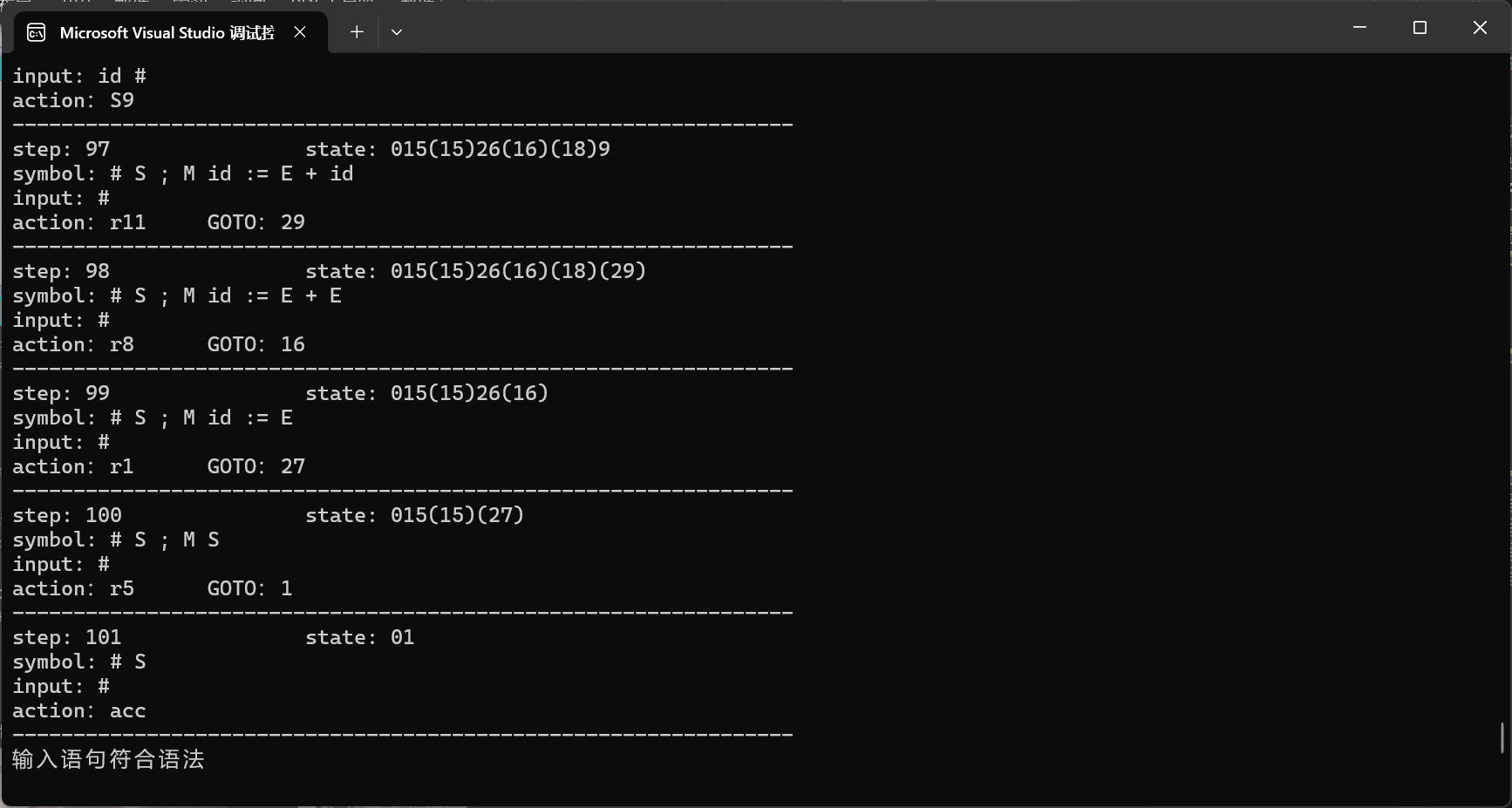


图4-3显示输入语句符合语法

4.4 测试用例4（测试用例3去掉一个;）

if True || False then a:=(b+c)\*a else a:=-b+c;

while b<=c && a<>c do b:=b+a

ans:=a+b+c;ans:=ans+d

#

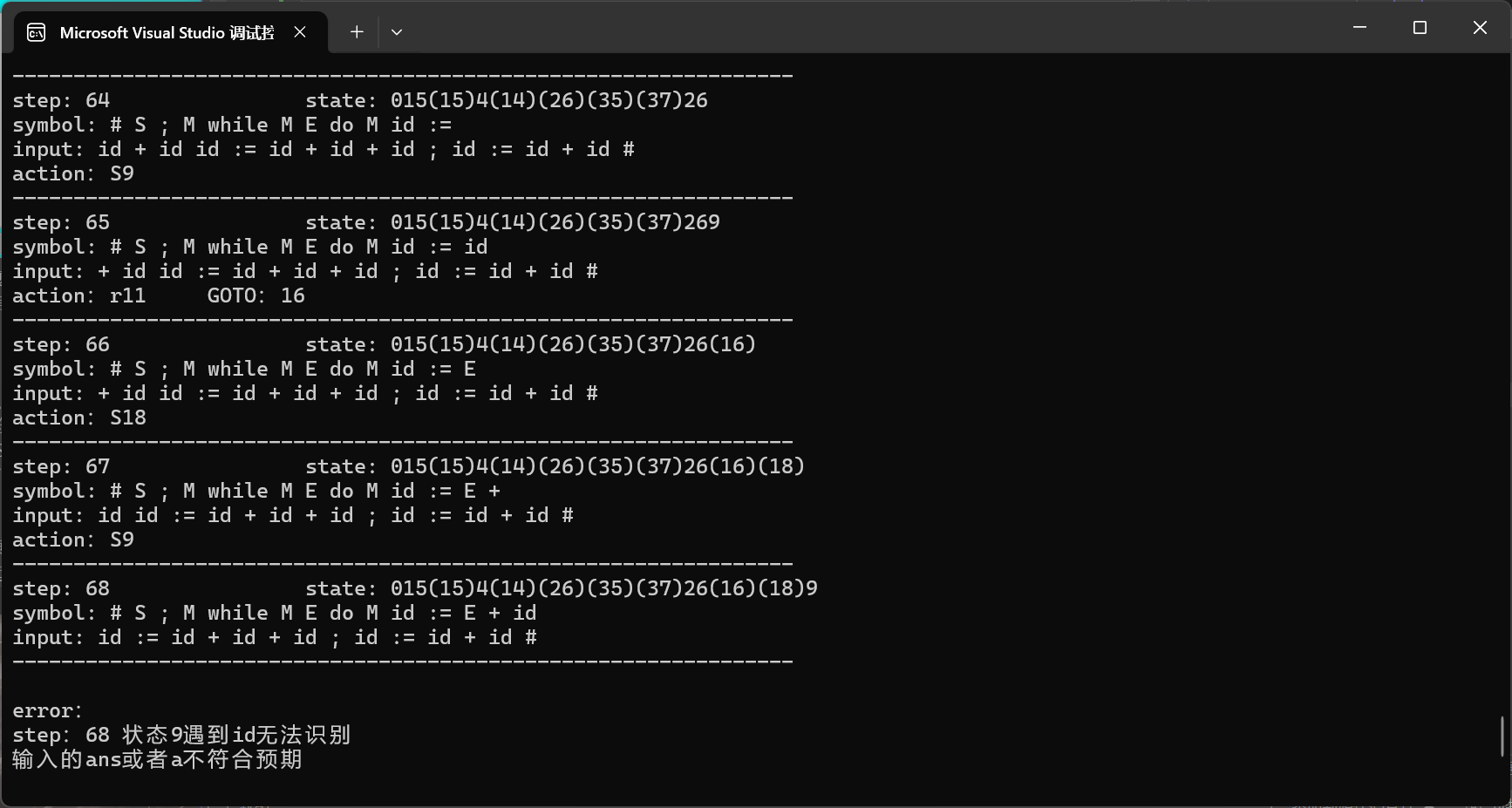


图4-4测试用例4-显示错误信息

在输出结果中发现原本应该时;的位置输入了不符合语法的ans，导致后序步骤无法进行。

4.55 测试用例5（测试用例3去掉#）

if True || False then a:=(b+c)\*a else a:=-b+c;

while b<=c && a<>c do b:=b+a;

ans:=a+b+c;ans:=ans+d

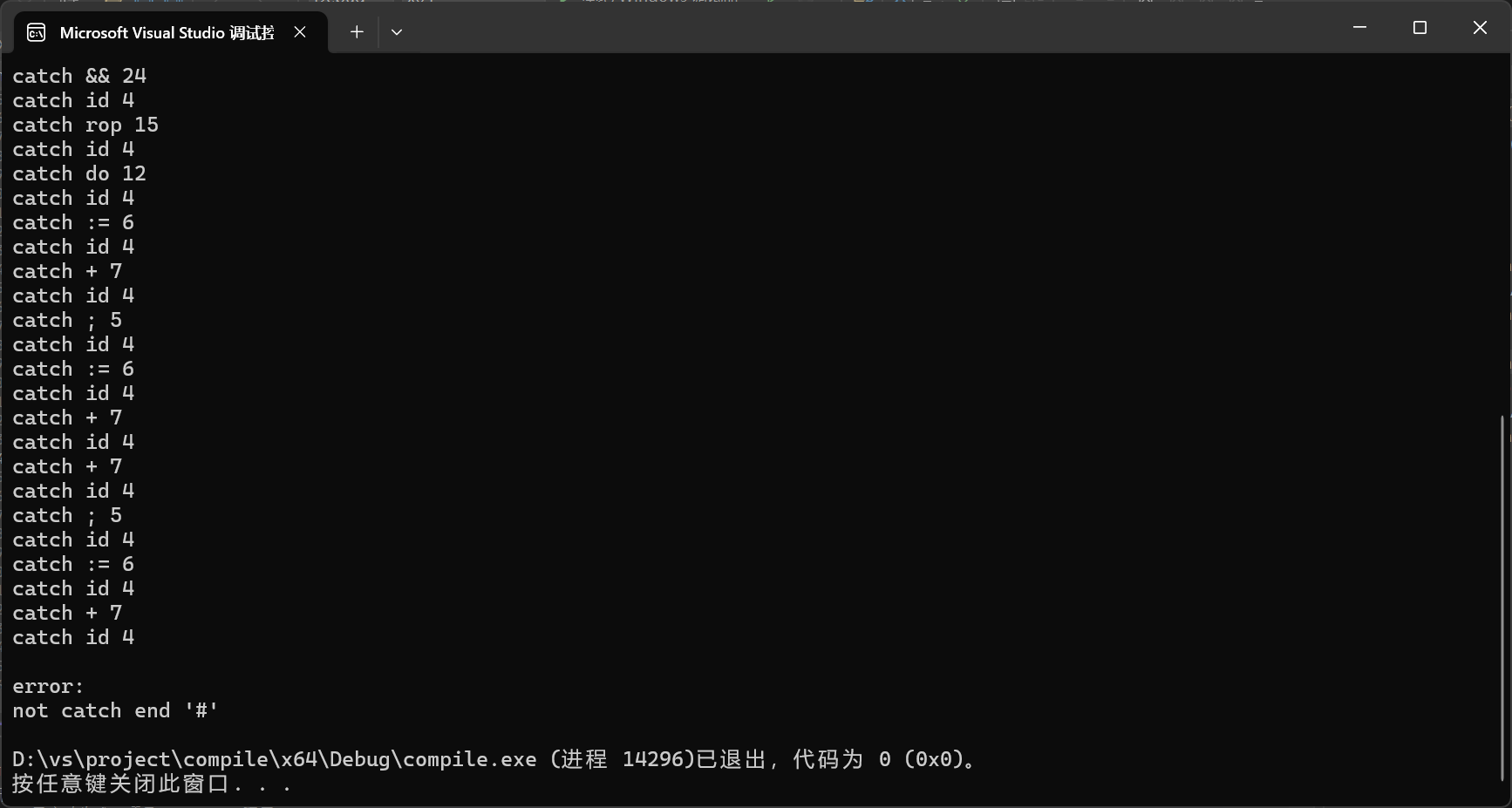


图4-5 测试用例5-显示错误信息

输入语句末尾没有#在词法分析阶段直接被判断为不符合词法而停止程序的继续执行。

5.2 语法分析遇到的问题与解决

语法分析遇到的第一个问题就是二义性，对于同时具有移进、规约操作的状态，初步可以根据相应的FOLLOW集合进行解决，但是仍然存在少量存在二义操作的状态，此时根据算符优先级以及左结合的原则可以将其消除。

其次，由于语法分析是与语义操作同时进行的，所以语法分析就不能像往常一样仅仅进行语法分析，而是要在进行语法分析的同时，向语义分析传递相应的参数，并对这些参数及时进行修改、维护以确保语义分析的正确性。

6 心得体会

此次编译原理课程设计加深了我使用自顶向下（递归下降）和自底向上（LR解析）的方法进行语法分析。这些技术可以帮助我正确解析输入，还让我能够构建解析树和抽象语法树（AST），为后续的语义分析和代码生成打下基础。通过不断调试和优化我的语法分析器，我体验到了设计简单而高效的文法所带来的挑战与乐趣。

同时，我的编程能力得到了显著提升。通过优化代码结构，我提高了代码的可读性，使得其他人能够更容易理解我的实现逻辑。而且，团队合作的经历让我明白了协作的价值。与同学们讨论语法规则和解析算法时，我学会了更好地表达自己的想法并注意聆听他人的见解，这使我们的代码质量得到了提升。在这个过程中，代码审查和相互支持也进一步激励了我的成长。