实验 1-2 词法语法分析器综合设计说明

[redacted]

一、运行和开发环境

①无图形界面：Win10 下 Visual Studio 开发，通过.exe 可执行文件在 cmd 窗口下运行。
②图形化界面：Win10 下 Qt5 开发，通过.exe 可执行文件由窗体程序运行。

二、功能

1、能识别的单词：

✓ 关键字：int | void | if | else | while | return
✓ 标识符： 字母（字母|数字）* （注：不与关键字相同）
✓ 数值：数字（数字）*
✓ 赋值号： =
✓ 算符： + | - | * | / | = | == | > | >= | < | <= | !=
✓ 界符： ;
✓ 分隔符： ,
✓ 注释号： /* */ | //
✓ 左括号： (
✓ 右括号：)
✓ 左大括号： {
✓ 右大括号： }
✓ 字母：| a |....| z | A |....| Z |
✓ 数字：0| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
✓ 结束符：#

扩充单词：

除上述作业要求单词以外，还扩充了如下单词：

✓ 关键字：char | const | unsigned | bool | true | false
✓ 左方括号：[
✓ 右方括号：]
✓ 单引号：’
✓ 双引号：”

2、能分析的文法：

本语法分析器主要采用 LL(1)文法：

✓ Program ::= <类型> < ID>’(‘ ’)’<语句块>
✓ <类型>::=int | void
✓ <ID>::=字母(字母|数字)*
✓ <语句块> ::= ‘{’ <内部声明> <语句串>’}’
✓ <内部声明> ::= 空 | <内部变量声明>{; <内部变量声明>}
✓ <内部变量声明>::=int <ID> （注： { }中的项表示可重复若干次）
✓ <语句串> ::= <语句> { <语句> }
✓ <语句> ::= <if 语句> |< while 语句> | <return 语句> | <赋值语句>
✓ <赋值语句> ::= <ID> =<表达式>;
✓ <return 语句> ::= return [<表达式>] （注：[ ]中的项表示可选）

✓ <while 语句> ::= while ‘(’ <表达式> ‘)’ <语句块>

✓ <if 语句> ::= if ‘(‘<表达式>’)’ <语句块> [else <语句块>]（注：[ ]中的项表示可选）

✓ <表达式>::=<加法表达式>{ relop <加法表达式> } （注：relop-> <|<=|>|>=|==|!=）

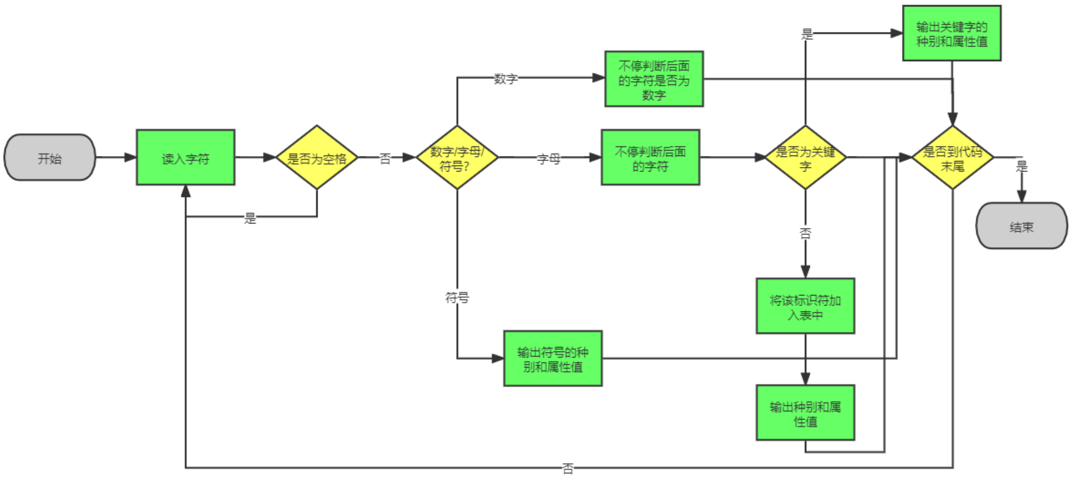
✓ <加法表达式> ::= <项> {+ <项> | -<项>}

✓ <项> ::= <因子> {* <因子> | /<因子>}

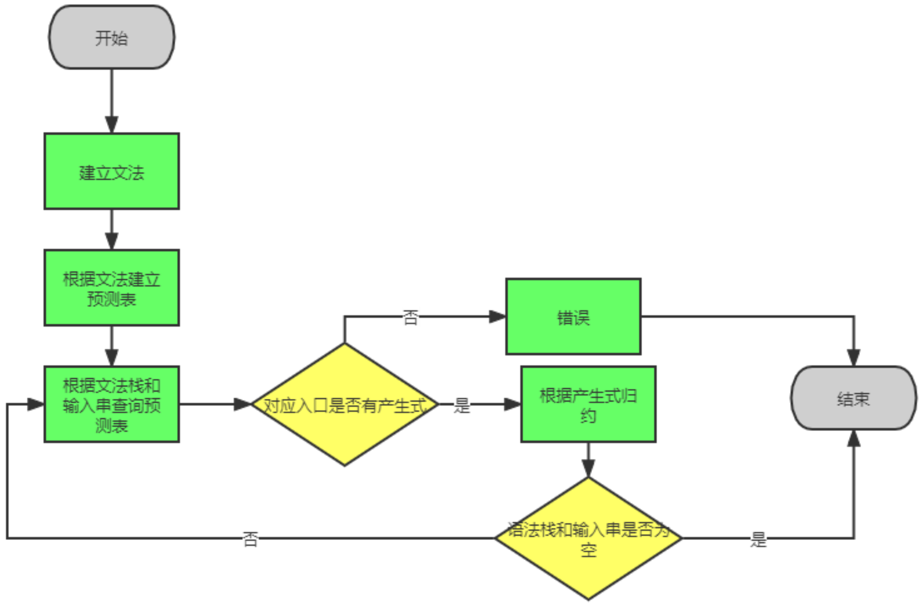
✓ <因子> ::=ID|num | ‘(’ <表达式>‘)’

三、主程序框图

词法分析器程序框图：

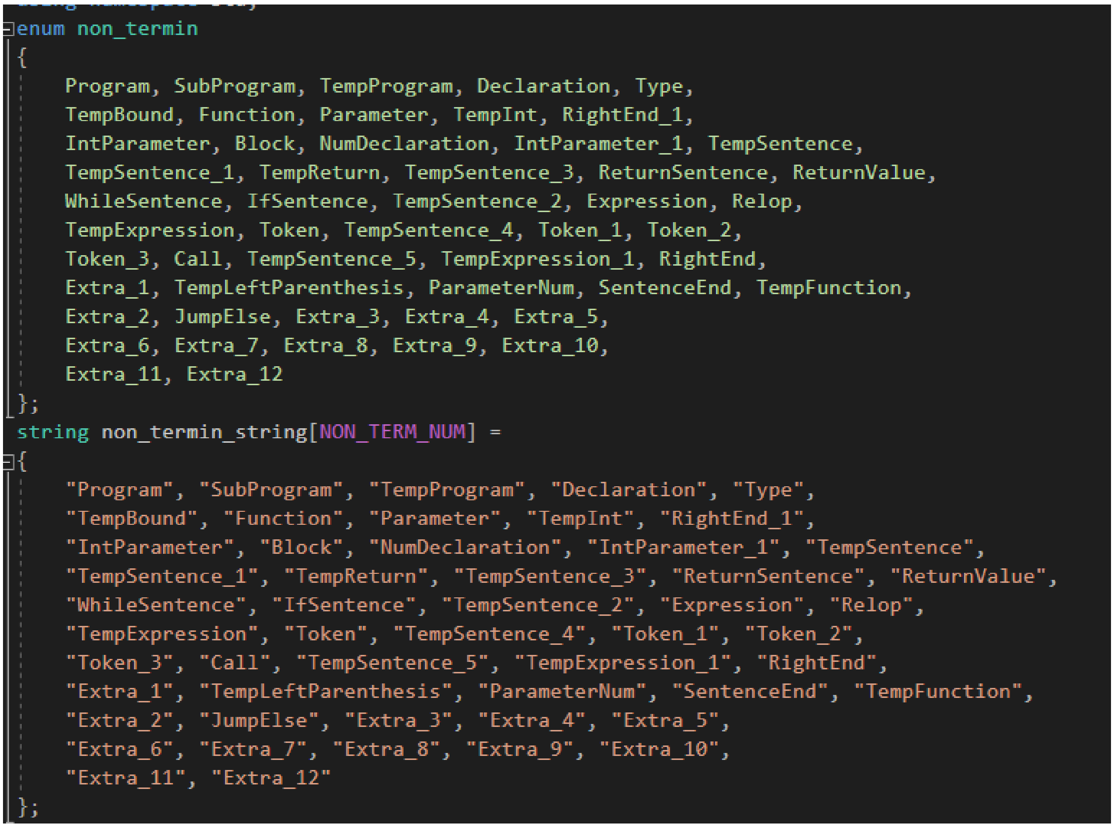


语法分析器程序框图：



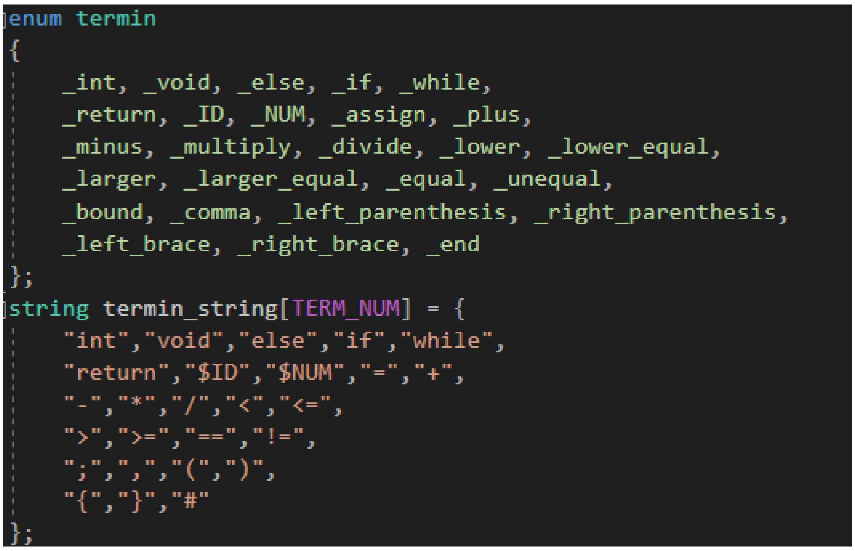
四、函数功能：

非终结符定义：



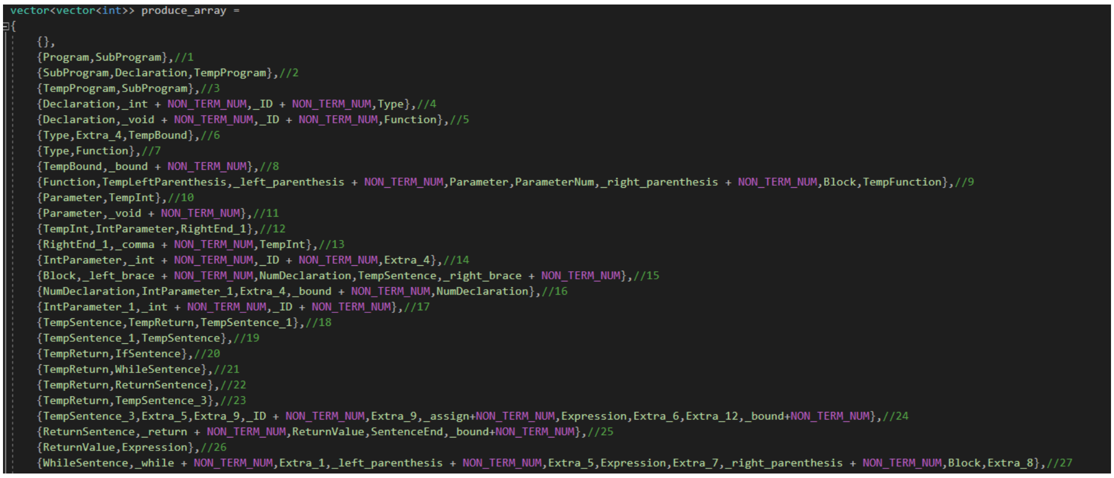
```
using namespace std;
enum non_termin
{
    Program, SubProgram, TempProgram, Declaration, Type,
    TempBound, Function, Parameter, TempInt, RightEnd_1,
    IntParameter, Block, NumDeclaration, IntParameter_1, TempSentence,
    TempSentence_1, TempReturn, TempSentence_3, ReturnSentence, ReturnValue,
    WhileSentence, IfSentence, TempSentence_2, Expression, Relop,
    TempExpression, Token, TempSentence_4, Token_1, Token_2,
    Token_3, Call, TempSentence_5, TempExpression_1, RightEnd,
    Extra_1, TempLeftParenthesis, ParameterNum, SentenceEnd, TempFunction,
    Extra_2, JumpElse, Extra_3, Extra_4, Extra_5,
    Extra_6, Extra_7, Extra_8, Extra_9, Extra_10,
    Extra_11, Extra_12
};
string non_termin_string[NON_TERM_NUM] =
{
    "Program", "SubProgram", "TempProgram", "Declaration", "Type",
    "TempBound", "Function", "Parameter", "TempInt", "RightEnd_1",
    "IntParameter", "Block", "NumDeclaration", "IntParameter_1", "TempSentence",
    "TempSentence_1", "TempReturn", "TempSentence_3", "ReturnSentence", "ReturnValue",
    "WhileSentence", "IfSentence", "TempSentence_2", "Expression", "Relop",
    "TempExpression", "Token", "TempSentence_4", "Token_1", "Token_2",
    "Token_3", "Call", "TempSentence_5", "TempExpression_1", "RightEnd",
    "Extra_1", "TempLeftParenthesis", "ParameterNum", "SentenceEnd", "TempFunction",
    "Extra_2", "JumpElse", "Extra_3", "Extra_4", "Extra_5",
    "Extra_6", "Extra_7", "Extra_8", "Extra_9", "Extra_10",
    "Extra_11", "Extra_12"
};
```

终结符定义:

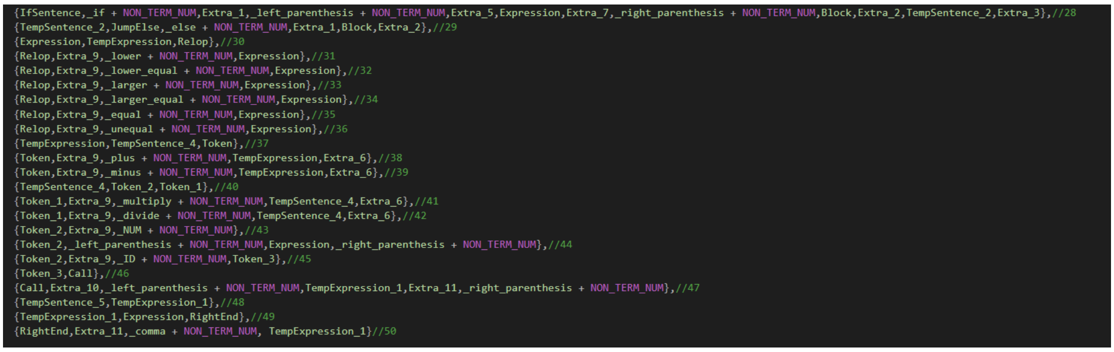


```
enum termin
{
    _int, _void, _else, _if, _while,
    _return, _ID, _NUM, _assign, _plus,
    _minus, _multiply, _divide, _lower, _lower_equal,
    _larger, _larger_equal, _equal, _unequal,
    _bound, _comma, _left_parenthesis, _right_parenthesis,
    _left_brace, _right_brace, _end
};
string termin_string[TERM_NUM] = {
    "int","void","else","if","while",
    "return","$ID","$NUM","=","+",
    "-","*","/","<","<=",
    ">",">=","==","!=",
    ";",",","(",")",
    "{","}","#"
};
```

产生式定义:

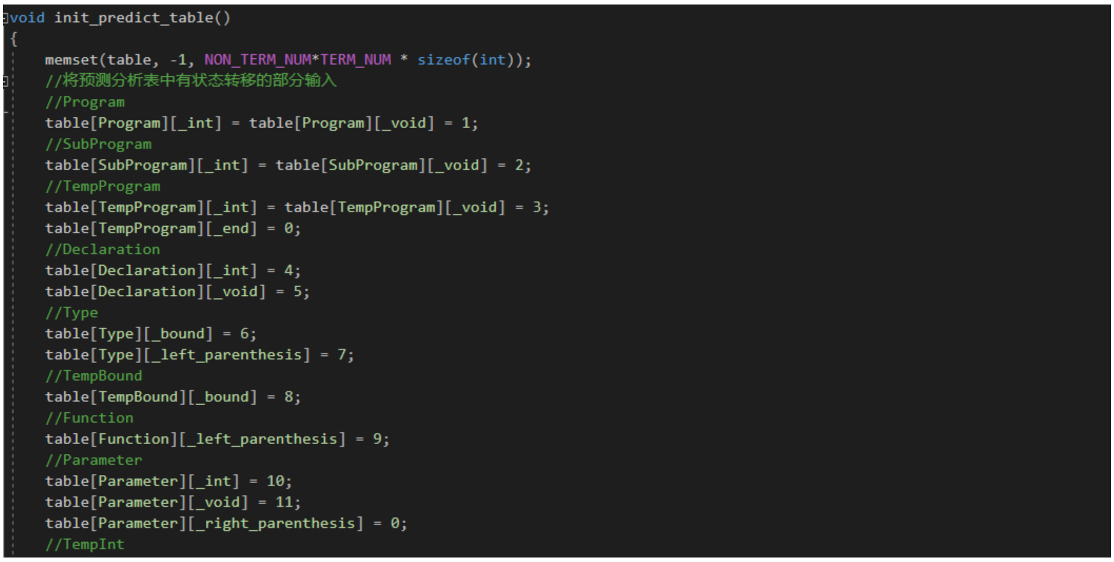


```
vector<vector<int>> produce_array =
{
    {},
    {Program,SubProgram},//1
    {SubProgram,Declaration,TempProgram},//2
    {TempProgram,SubProgram},//3
    {Declaration,_int + NON_TERM_NUM,_ID + NON_TERM_NUM,Type},//4
    {Declaration,_void + NON_TERM_NUM,_ID + NON_TERM_NUM,Function},//5
    {Type,Extra_4,TempBound},//6
    {Type,Function},//7
    {TempBound,_bound + NON_TERM_NUM},//8
    {Function,TempLeftParenthesis,_left_parenthesis + NON_TERM_NUM,Parameter,ParameterNum,_right_parenthesis + NON_TERM_NUM,Block,TempFunction},//9
    {Parameter,TempInt},//10
    {Parameter,_void + NON_TERM_NUM},//11
    {TempInt,IntParameter,RightEnd_1},//12
    {RightEnd_1,_comma + NON_TERM_NUM,TempInt},//13
    {IntParameter,_int + NON_TERM_NUM,_ID + NON_TERM_NUM,Extra_4},//14
    {Block,_left_brace + NON_TERM_NUM,NumDeclaration,TempSentence,_right_brace + NON_TERM_NUM},//15
    {NumDeclaration,IntParameter_1,Extra_4,_bound + NON_TERM_NUM,NumDeclaration},//16
    {IntParameter_1,_int + NON_TERM_NUM,_ID + NON_TERM_NUM},//17
    {TempSentence,TempReturn,TempSentence_1},//18
    {TempSentence_1,TempSentence},//19
    {TempReturn,IfSentence},//20
    {TempReturn,WhileSentence},//21
    {TempReturn,ReturnSentence},//22
    {TempReturn,TempSentence_3},//23
    {TempSentence_3,Extra_5,Extra_9,_ID + NON_TERM_NUM,Extra_9,_assign+NON_TERM_NUM,Expression,Extra_6,Extra_12,_bound+NON_TERM_NUM},//24
    {ReturnSentence,_return + NON_TERM_NUM,ReturnValue,SentenceEnd,_bound+NON_TERM_NUM},//25
    {ReturnValue,Expression},//26
    {WhileSentence,_while + NON_TERM_NUM,Extra_1,_left_parenthesis + NON_TERM_NUM,Extra_5,Expression,Extra_7,_right_parenthesis + NON_TERM_NUM,Block,Extra_8},//27
```



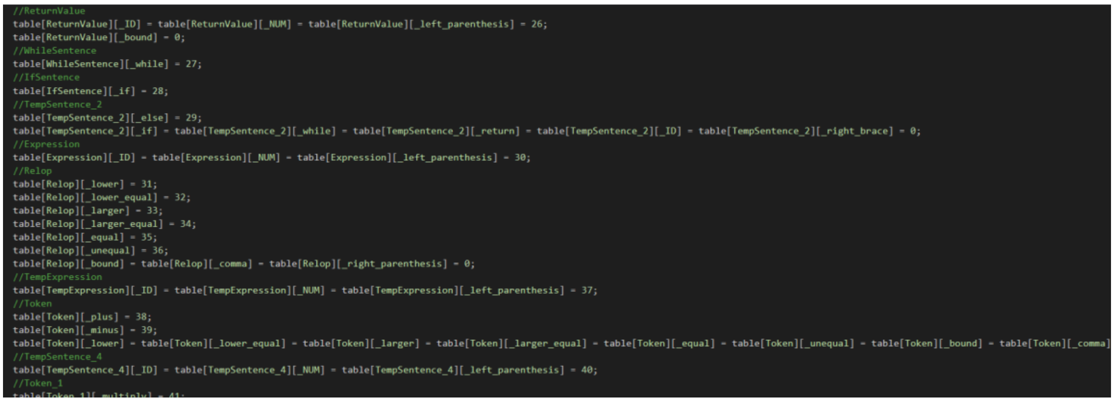
```
{IfSentence,_if + NON_TERM_NUM,Extra_1,_left_parenthesis + NON_TERM_NUM,Extra_5,Expression,Extra_7,_right_parenthesis + NON_TERM_NUM,Block,Extra_2,TempSentence_2,Extra_3},//28
{TempSentence_2,JumpElse,_else + NON_TERM_NUM,Extra_1,Block,Extra_2},//29
{Expression,TempExpression,Relop},//30
{Relop,Extra_9,_lower + NON_TERM_NUM,Expression},//31
{Relop,Extra_9,_lower_equal + NON_TERM_NUM,Expression},//32
{Relop,Extra_9,_larger + NON_TERM_NUM,Expression},//33
{Relop,Extra_9,_larger_equal + NON_TERM_NUM,Expression},//34
{Relop,Extra_9,_equal + NON_TERM_NUM,Expression},//35
{Relop,Extra_9,_unequal + NON_TERM_NUM,Expression},//36
{TempExpression,TempSentence_4,Token},//37
{Token,Extra_9,_plus + NON_TERM_NUM,TempExpression,Extra_6},//38
{Token,Extra_9,_minus + NON_TERM_NUM,TempExpression,Extra_6},//39
{TempSentence_4,Token_2,Token_1},//40
{Token_1,Extra_9,_multiply + NON_TERM_NUM,TempSentence_4,Extra_6},//41
{Token_1,Extra_9,_divide + NON_TERM_NUM,TempSentence_4,Extra_6},//42
{Token_2,Extra_9,_NUM + NON_TERM_NUM},//43
{Token_2,_left_parenthesis + NON_TERM_NUM,Expression,_right_parenthesis + NON_TERM_NUM},//44
{Token_2,Extra_9,_ID + NON_TERM_NUM,Token_3},//45
{Token_3,Call},//46
{Call,Extra_10,_left_parenthesis + NON_TERM_NUM,TempExpression_1,Extra_11,_right_parenthesis + NON_TERM_NUM},//47
{TempSentence_5,TempExpression_1},//48
{TempExpression_1,Expression,RightEnd},//49
{RightEnd,Extra_11,_comma + NON_TERM_NUM, TempExpression_1}//50
```

预测表定义：



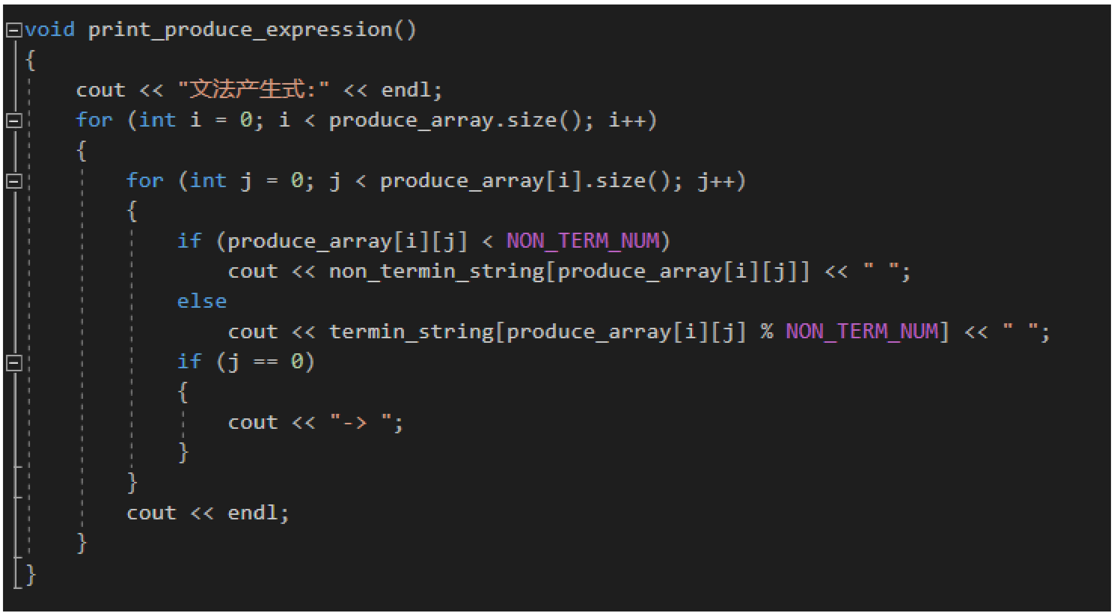
```
void init_predict_table()
{
    memset(table, -1, NON_TERM_NUM*TERM_NUM * sizeof(int));
    //将预测分析表中有状态转移的部分输入
    //Program
    table[Program][_int] = table[Program][_void] = 1;
    //SubProgram
    table[SubProgram][_int] = table[SubProgram][_void] = 2;
    //TempProgram
    table[TempProgram][_int] = table[TempProgram][_void] = 3;
    table[TempProgram][_end] = 0;
    //Declaration
    table[Declaration][_int] = 4;
    table[Declaration][_void] = 5;
    //Type
    table[Type][_bound] = 6;
    table[Type][_left_parenthesis] = 7;
    //TempBound
    table[TempBound][_bound] = 8;
    //Function
    table[Function][_left_parenthesis] = 9;
    //Parameter
    table[Parameter][_int] = 10;
    table[Parameter][_void] = 11;
    table[Parameter][_right_parenthesis] = 0;
    //TempInt
```

```
//TempInt
table[TempInt][_int] = 12;
//RightEnd_1
table[RightEnd_1][_comma] = 13;
table[RightEnd_1][_right_parenthesis] = 0;
//IntParameter
table[IntParameter][_int] = 14;
//Block
table[Block][_left_brace] = 15;
//NumDeclaration
table[NumDeclaration][_int] = 16;
table[NumDeclaration][_if] = table[NumDeclaration][_while] = table[NumDeclaration][_return] = table[NumDeclaration][_ID] = 0;
//IntParameter_1
table[IntParameter_1][_int] = 17;
//TempSentence
table[TempSentence][_if] = table[TempSentence][_while] = table[TempSentence][_return] = table[TempSentence][_ID] = 18;
//TempSentence_1
table[TempSentence_1][_if] = table[TempSentence_1][_while] = table[TempSentence_1][_return] = table[TempSentence_1][_ID] = 19;
table[TempSentence_1][_right_brace] = 0;
//TempReturn
table[TempReturn][_if] = 20;
table[TempReturn][_while] = 21;
table[TempReturn][_return] = 22;
table[TempReturn][_ID] = 23;
//TempSentence_3
table[TempSentence_3][_ID] = 24;
//ReturnSentence
table[ReturnSentence][_return] = 25;
```



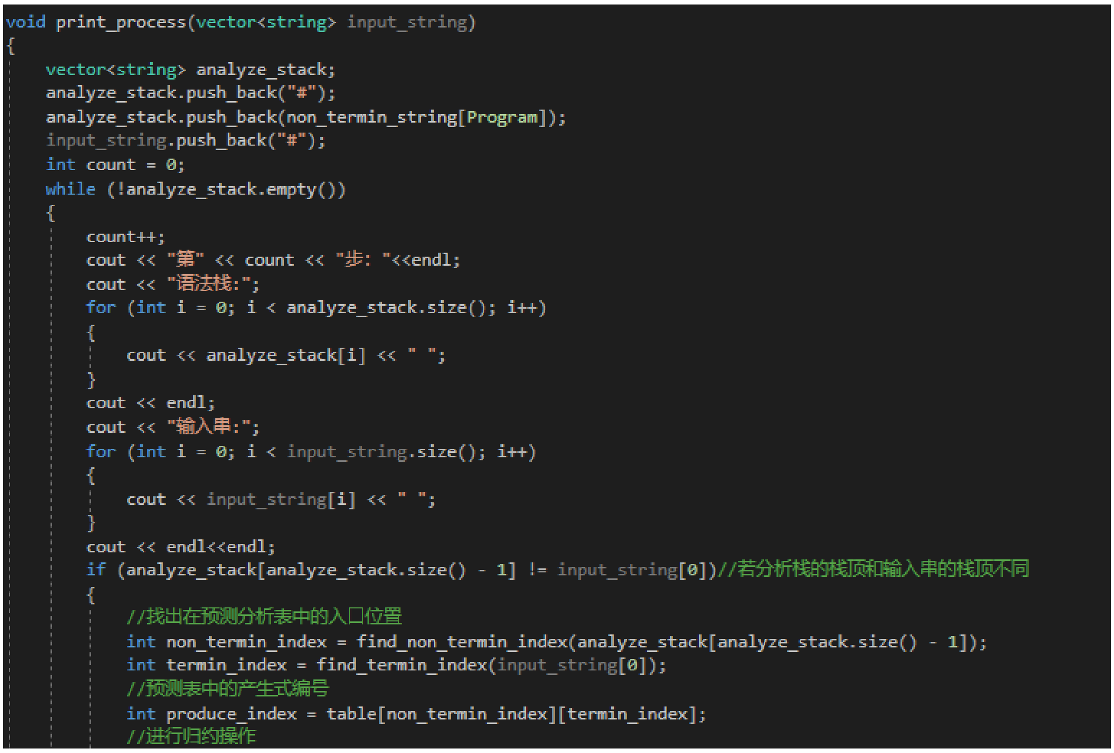
```
//ReturnValue
table[ReturnValue][_ID] = table[ReturnValue][_NUM] = table[ReturnValue][_left_parenthesis] = 26;
table[ReturnValue][_bound] = 0;
//WhileSentence
table[WhileSentence][_while] = 27;
//IfSentence
table[IfSentence][_if] = 28;
//TempSentence_2
table[TempSentence_2][_else] = 29;
table[TempSentence_2][_if] = table[TempSentence_2][_while] = table[TempSentence_2][_return] = table[TempSentence_2][_ID] = table[TempSentence_2][_right_brace] = 0;
//Expression
table[Expression][_ID] = table[Expression][_NUM] = table[Expression][_left_parenthesis] = 30;
//Relop
table[Relop][_lower] = 31;
table[Relop][_lower_equal] = 32;
table[Relop][_larger] = 33;
table[Relop][_larger_equal] = 34;
table[Relop][_equal] = 35;
table[Relop][_unequal] = 36;
table[Relop][_bound] = table[Relop][_comma] = table[Relop][_right_parenthesis] = 0;
//TempExpression
table[TempExpression][_ID] = table[TempExpression][_NUM] = table[TempExpression][_left_parenthesis] = 37;
//Token
table[Token][_plus] = 38;
table[Token][_minus] = 39;
table[Token][_lower] = table[Token][_lower_equal] = table[Token][_larger] = table[Token][_larger_equal] = table[Token][_equal] = table[Token][_unequal] = table[Token][_bound] = table[Token][_comma]
//TempSentence_4
table[TempSentence_4][_ID] = table[TempSentence_4][_NUM] = table[TempSentence_4][_left_parenthesis] = 40;
//Token_1
table[Token_1][_multiply] = 41;
```

产生式打印:

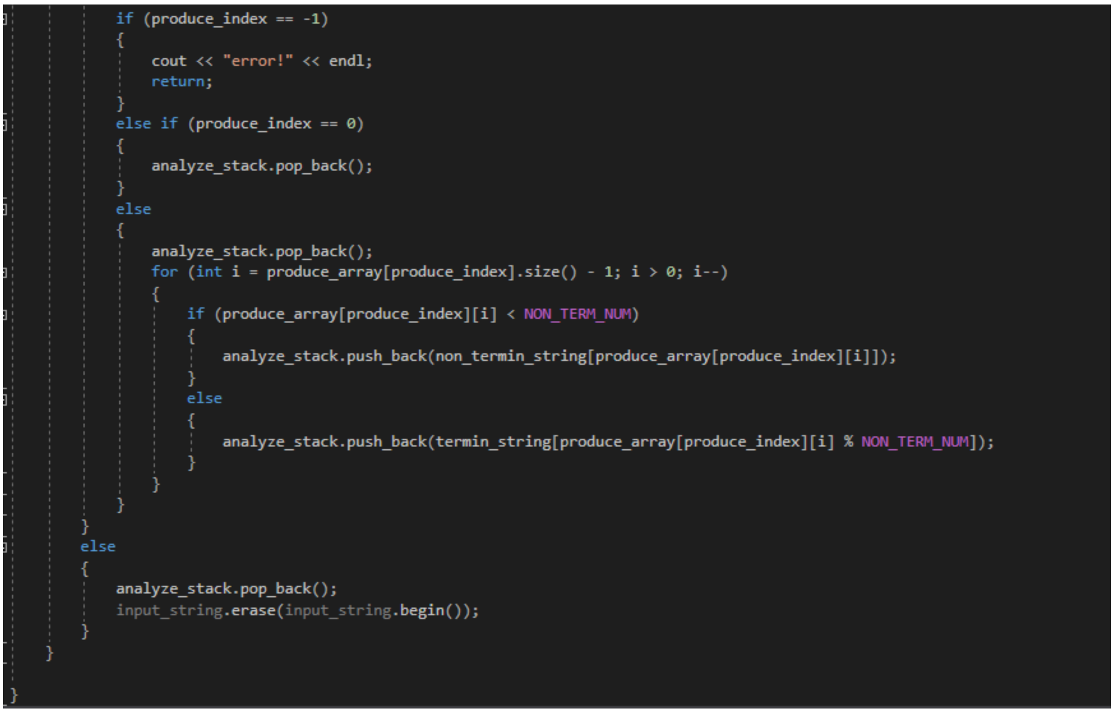


```
void print_produce_expression()
{
    cout << "文法产生式:" << endl;
    for (int i = 0; i < produce_array.size(); i++)
    {
        for (int j = 0; j < produce_array[i].size(); j++)
        {
            if (produce_array[i][j] < NON_TERM_NUM)
                cout << non_termin_string[produce_array[i][j]] << " ";
            else
                cout << termin_string[produce_array[i][j] % NON_TERM_NUM] << " ";
            if (j == 0)
            {
                cout << "-> ";
            }
        }
        cout << endl;
    }
}
```

归约过程:



```cpp
void print_process(vector<string> input_string)
{
    vector<string> analyze_stack;
    analyze_stack.push_back("#");
    analyze_stack.push_back(non_termin_string[Program]);
    input_string.push_back("#");
    int count = 0;
    while (!analyze_stack.empty())
    {
        count++;
        cout << "第" << count << "步: "<<endl;
        cout << "语法栈:";
        for (int i = 0; i < analyze_stack.size(); i++)
        {
            cout << analyze_stack[i] << " ";
        }
        cout << endl;
        cout << "输入串:";
        for (int i = 0; i < input_string.size(); i++)
        {
            cout << input_string[i] << " ";
        }
        cout << endl<<endl;
        if (analyze_stack[analyze_stack.size() - 1] != input_string[0])//若分析栈的栈顶和输入串的栈顶不同
        {
            //找出在预测分析表中的入口位置
            int non_termin_index = find_non_termin_index(analyze_stack[analyze_stack.size() - 1]);
            int termin_index = find_termin_index(input_string[0]);
            //预测表中的产生式编号
            int produce_index = table[non_termin_index][termin_index];
            //进行归约操作
```

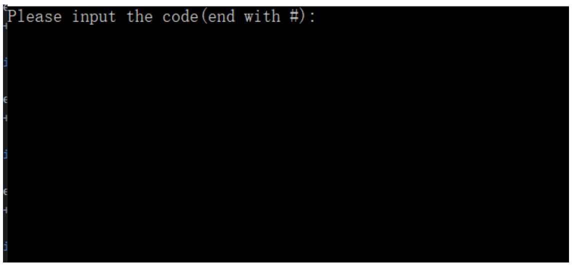


```cpp
            if (produce_index == -1)
            {
                cout << "error!" << endl;
                return;
            }
            else if (produce_index == 0)
            {
                analyze_stack.pop_back();
            }
            else
            {
                analyze_stack.pop_back();
                for (int i = produce_array[produce_index].size() - 1; i > 0; i--)
                {
                    if (produce_array[produce_index][i] < NON_TERM_NUM)
                    {
                        analyze_stack.push_back(non_termin_string[produce_array[produce_index][i]]);
                    }
                    else
                    {
                        analyze_stack.push_back(termin_string[produce_array[produce_index][i] % NON_TERM_NUM]);
                    }
                }
            }
        }
        else
        {
            analyze_stack.pop_back();
            input_string.erase(input_string.begin());
        }
    }
}
```

四、运行结果

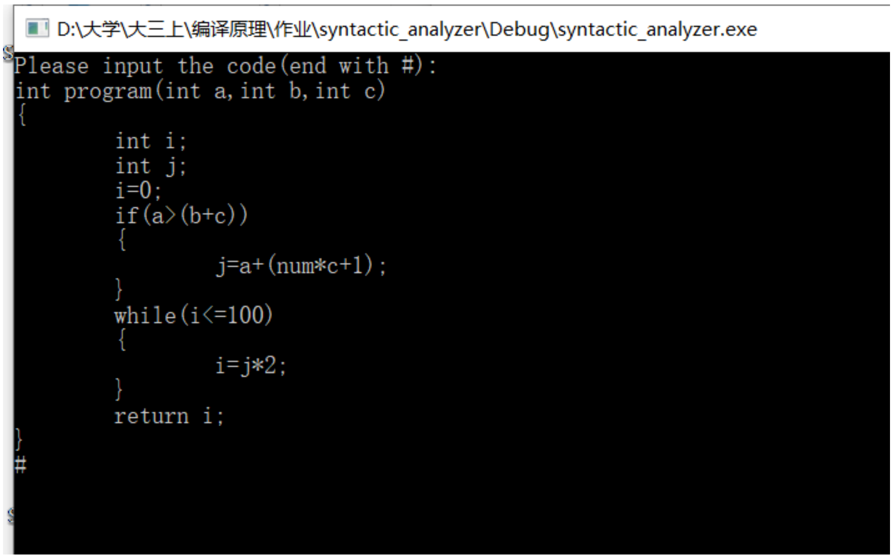
①无图形界面

没有使用图形界面，打开可执行文件（作业目录下的 syntactic_analyzer_console.exe），可以看见输入代码的提示。



```
Please input the code(end with #):
```

通过控制台键盘输入代码，另起一行顶格输入#结束。（同目录下有 test.txt 的测试用例）

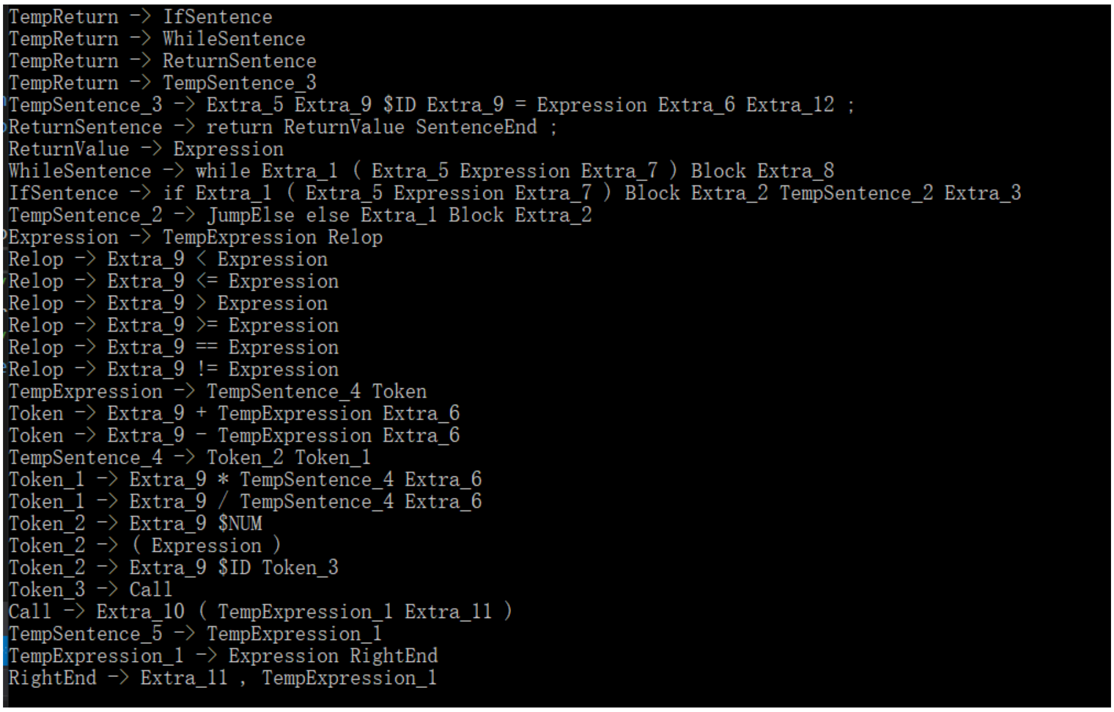


D:\大学\大三上\编译原理\作业\syntactic_analyzer\Debug\syntactic_analyzer.exe

```
Please input the code(end with #):
int program(int a,int b,int c)
{
        int i;
        int j;
        i=0;
        if(a>(b+c))
        {
                j=a+(num*c+1);
        }
        while(i<=100)
        {
                i=j*2;
        }
        return i;
}
#
```

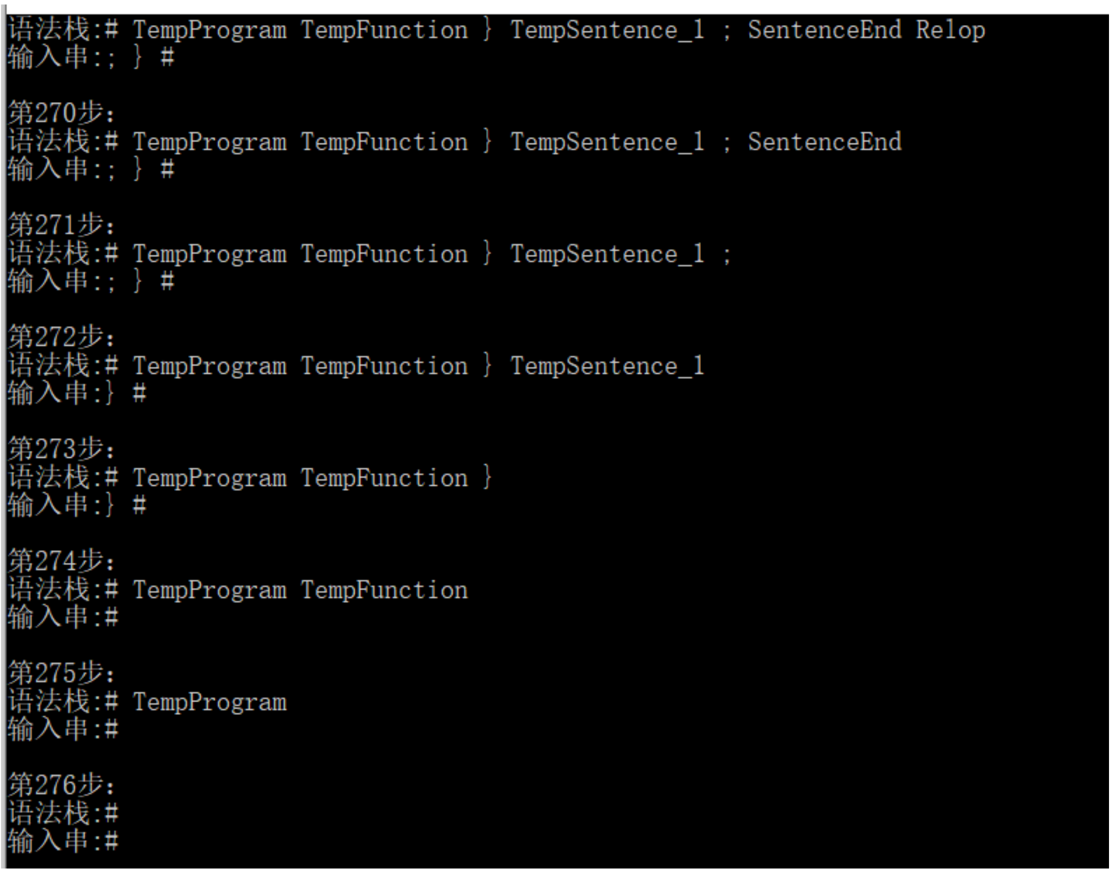
输出结果如下所示：

文法产生式：



```
TempReturn -> IfSentence
TempReturn -> WhileSentence
TempReturn -> ReturnSentence
TempReturn -> TempSentence_3
TempSentence_3 -> Extra_5 Extra_9 $ID Extra_9 = Expression Extra_6 Extra_12 ;
ReturnSentence -> return ReturnValue SentenceEnd ;
ReturnValue -> Expression
WhileSentence -> while Extra_1 ( Extra_5 Expression Extra_7 ) Block Extra_8
IfSentence -> if Extra_1 ( Extra_5 Expression Extra_7 ) Block Extra_2 TempSentence_2 Extra_3
TempSentence_2 -> JumpElse else Extra_1 Block Extra_2
Expression -> TempExpression Relop
Relop -> Extra_9 < Expression
Relop -> Extra_9 <= Expression
Relop -> Extra_9 > Expression
Relop -> Extra_9 >= Expression
Relop -> Extra_9 == Expression
Relop -> Extra_9 != Expression
TempExpression -> TempSentence_4 Token
Token -> Extra_9 + TempExpression Extra_6
Token -> Extra_9 - TempExpression Extra_6
TempSentence_4 -> Token_2 Token_1
Token_1 -> Extra_9 * TempSentence_4 Extra_6
Token_1 -> Extra_9 / TempSentence_4 Extra_6
Token_2 -> Extra_9 $NUM
Token_2 -> ( Expression )
Token_2 -> Extra_9 $ID Token_3
Token_3 -> Call
Call -> Extra_10 ( TempExpression_1 Extra_11 )
TempSentence_5 -> TempExpression_1
TempExpression_1 -> Expression RightEnd
RightEnd -> Extra_11 , TempExpression_1
```

归约过程：



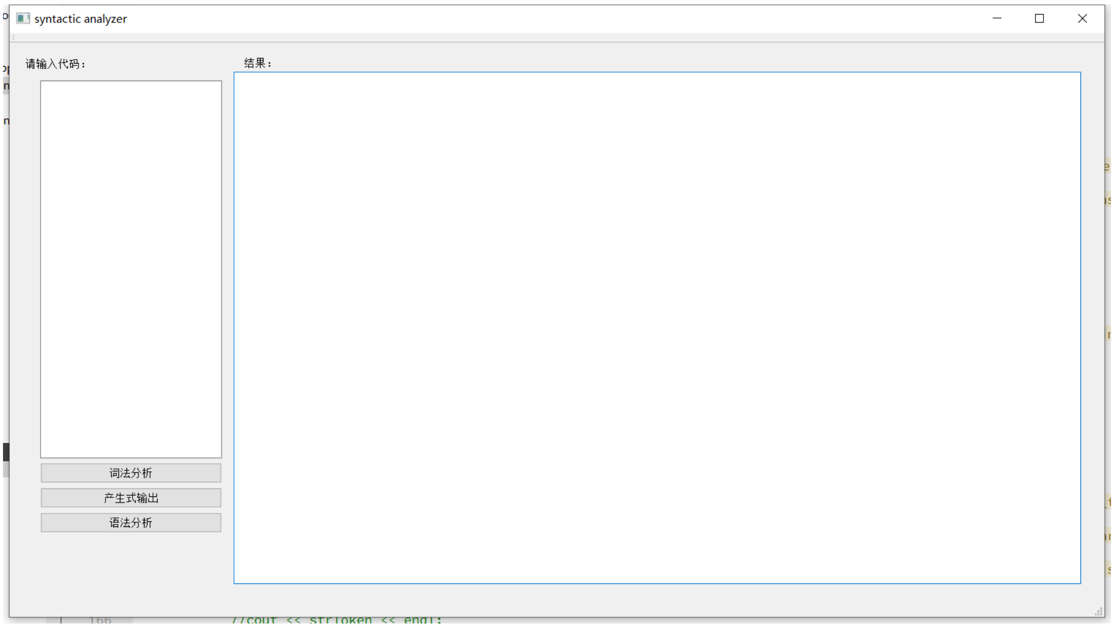
```
语法栈:# TempProgram TempFunction } TempSentence_1 ; SentenceEnd Relop
输入串:; } #

第270步:
语法栈:# TempProgram TempFunction } TempSentence_1 ; SentenceEnd
输入串:; } #

第271步:
语法栈:# TempProgram TempFunction } TempSentence_1 ;
输入串:; } #

第272步:
语法栈:# TempProgram TempFunction } TempSentence_1
输入串:} #

第273步:
语法栈:# TempProgram TempFunction }
输入串:} #

第274步:
语法栈:# TempProgram TempFunction
输入串:#

第275步:
语法栈:# TempProgram
输入串:#

第276步:
语法栈:#
输入串:#
```

②图形化界面

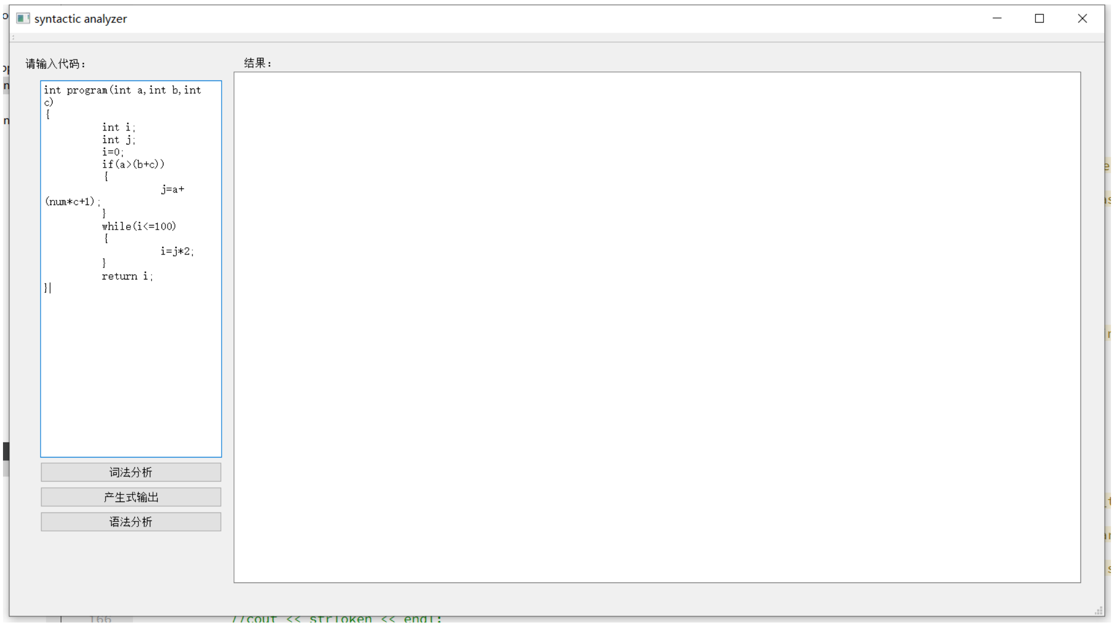
打开可执行文件 syntactic_anlayzer_gui（作业目录下的 syntactic_analyzer_gui/syntactic_analyzer_gui.exe），可运行图形化的词法分析器。

图形化界面如图所示

左半部分输入代码，右半部分输出结果。

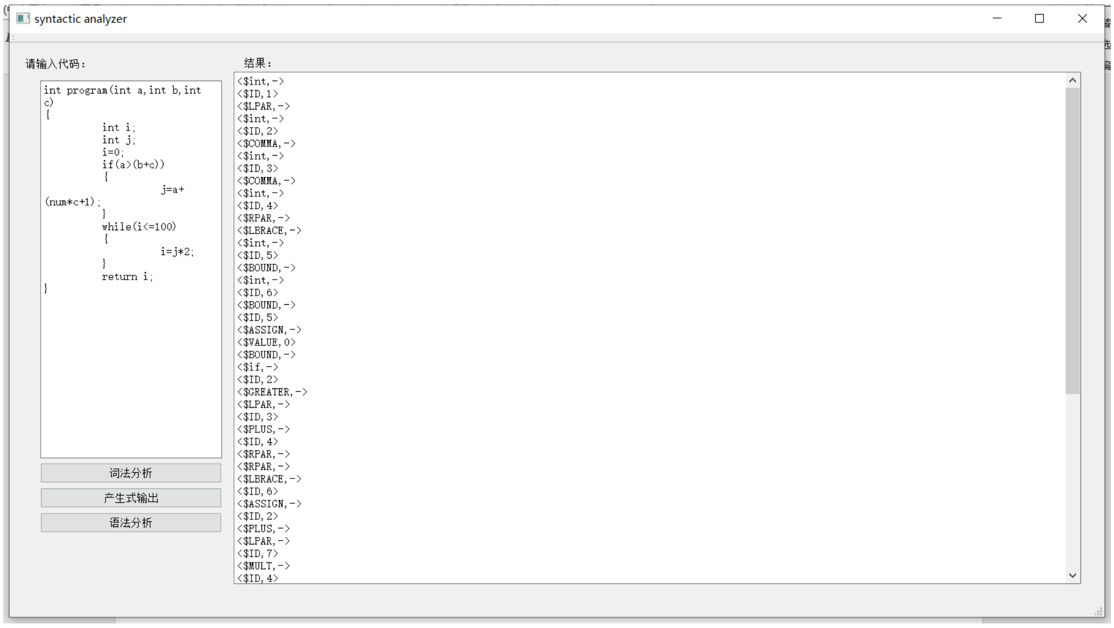


输入代码:



```
int program(int a,int b,int
c)
{
          int i;
          int j;
          i=0;
          if(a>(b+c))
          {
                    j=a+
(num*c+1);
          }
          while(i<=100)
          {
                    i=j*2;
          }
          return i;
}|
```

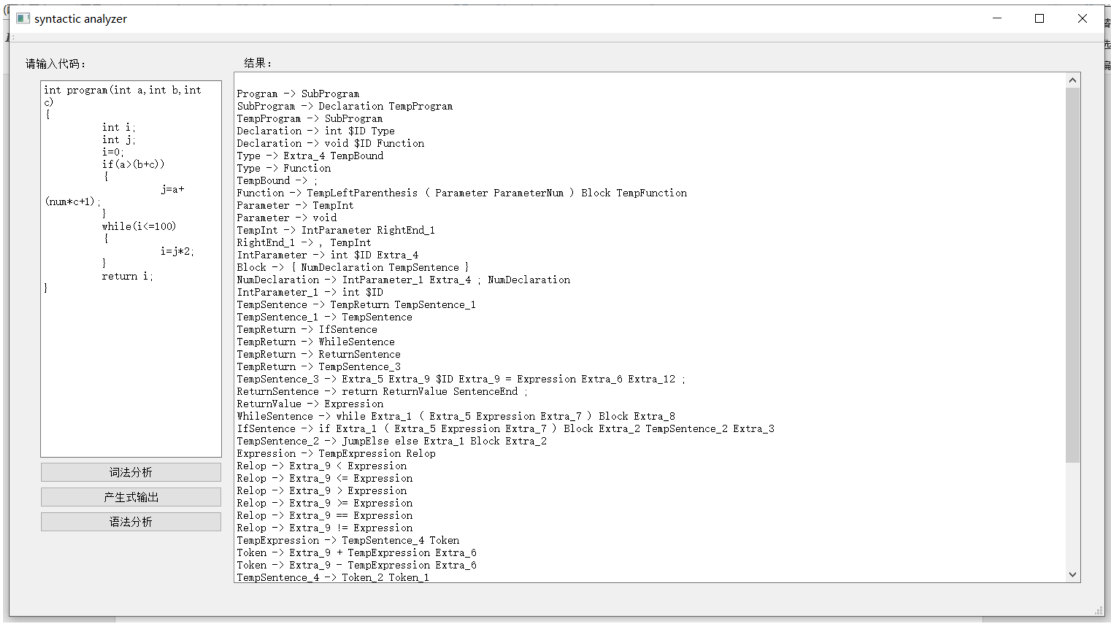
按下“词法分析”的按钮，右方输出词法分析结果。



```
int program(int a,int b,int
c)
{
          int i;
          int j;
          i=0;
          if(a>(b+c))
          {
                    j=a+
(num*c+1);
          }
          while(i<=100)
          {
                    i=j*2;
          }
          return i;
}
```

```
<$int,->
<$ID,1>
<$LPAR,->
<$int,->
<$ID,2>
<$COMMA,->
<$int,->
<$ID,3>
<$COMMA,->
<$int,->
<$ID,4>
<$RPAR,->
<$LBRACE,->
<$int,->
<$ID,5>
<$BOUND,->
<$int,->
<$ID,6>
<$BOUND,->
<$ID,5>
<$ASSIGN,->
<$VALUE,0>
<$BOUND,->
<$if,->
<$ID,2>
<$GREATER,->
<$LPAR,->
<$ID,3>
<$PLUS,->
<$ID,4>
<$RPAR,->
<$RPAR,->
<$LBRACE,->
<$ID,6>
<$ASSIGN,->
<$ID,2>
<$PLUS,->
<$LPAR,->
<$ID,7>
<$MULT,->
<$ID,4>
```

按下“产生式输出”的按钮，右方输出产生式。

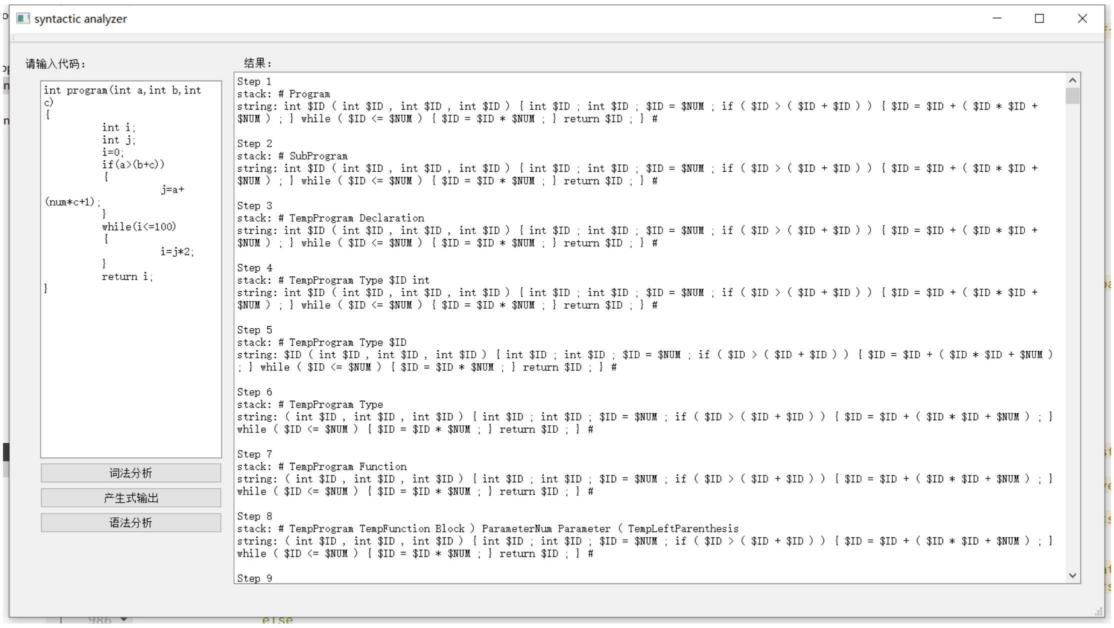


```
int program(int a,int b,int
c)
{
        int i;
        int j;
        i=0;
        if(a>(b+c))
        {
                j=a+
(num*c+1);
        }
        while(i<=100)
        {
                i=j*2;
        }
        return i;
}
```

```
Program -> SubProgram
SubProgram -> Declaration TempProgram
TempProgram -> SubProgram
Declaration -> int $ID Type
Declaration -> void $ID Function
Type -> Extra_4 TempBound
Type -> Function
TempBound -> ;
Function -> TempLeftParenthesis ( Parameter ParameterNum ) Block TempFunction
Parameter -> TempInt
Parameter -> void
TempInt -> IntParameter RightEnd_1
RightEnd_1 -> , TempInt
IntParameter -> int $ID Extra_4
Block -> { NumDeclaration TempSentence }
NumDeclaration -> IntParameter_1 Extra_4 ; NumDeclaration
IntParameter_1 -> int $ID
TempSentence -> TempReturn TempSentence_1
TempSentence_1 -> TempSentence
TempReturn -> IfSentence
TempReturn -> WhileSentence
TempReturn -> ReturnSentence
TempReturn -> TempSentence_3
TempSentence_3 -> Extra_5 Extra_9 $ID Extra_9 = Expression Extra_6 Extra_12 ;
ReturnSentence -> return ReturnValue SentenceEnd ;
ReturnValue -> Expression
WhileSentence -> while Extra_1 ( Extra_5 Expression Extra_7 ) Block Extra_8
IfSentence -> if Extra_1 ( Extra_5 Expression Extra_7 ) Block Extra_2 TempSentence_2 Extra_3
TempSentence_2 -> JumpElse else Extra_1 Block Extra_2
Expression -> TempExpression Relop
Relop -> Extra_9 < Expression
Relop -> Extra_9 <= Expression
Relop -> Extra_9 > Expression
Relop -> Extra_9 >= Expression
Relop -> Extra_9 == Expression
Relop -> Extra_9 != Expression
TempExpression -> TempSentence_4 Token
Token -> Extra_9 + TempExpression Extra_6
Token -> Extra_9 - TempExpression Extra_6
TempSentence_4 -> Token_2 Token_1
```

按下“语法分析”的按钮，右方输出语法分析。

语法分析打印内容：步骤、语法栈和输入串中内容。

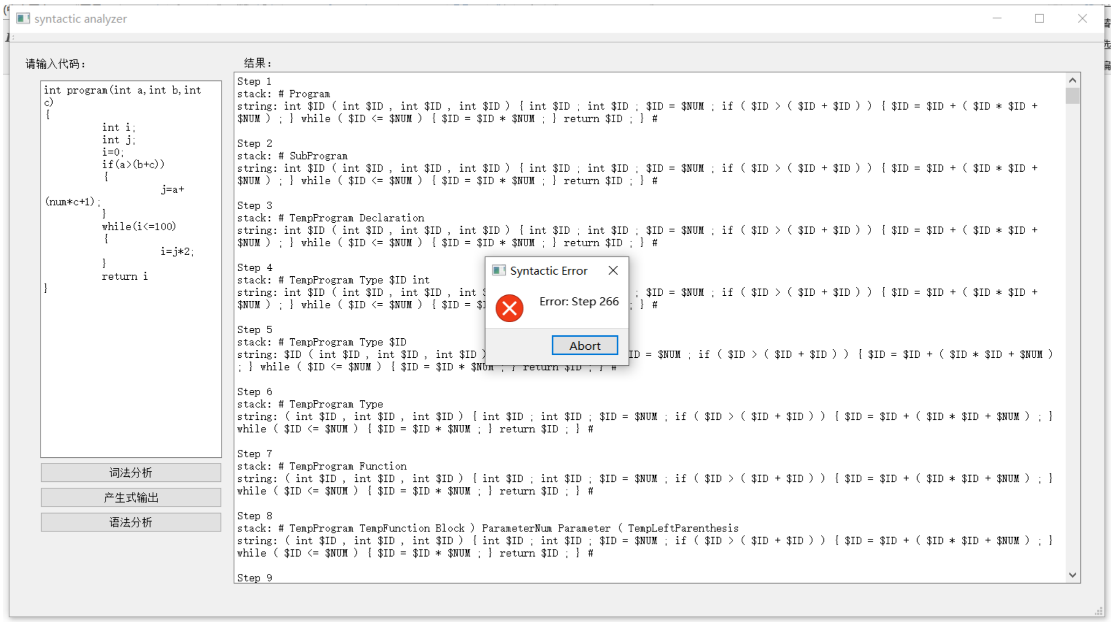


```
int program(int a,int b,int
c)
{
        int i;
        int j;
        i=0;
        if(a>(b+c))
        {
                j=a+
(num*c+1);
        }
        while(i<=100)
        {
                i=j*2;
        }
        return i;
}
```

```
Step 1
stack: # Program
string: int $ID ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID +
$NUM ) ; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 2
stack: # SubProgram
string: int $ID ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID +
$NUM ) ; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 3
stack: # TempProgram Declaration
string: int $ID ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID +
$NUM ) ; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 4
stack: # TempProgram Type $ID int
string: int $ID ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID +
$NUM ) ; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 5
stack: # TempProgram Type $ID
string: $ID ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID + $NUM )
; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 6
stack: # TempProgram Type
string: ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID + $NUM ) ; }
while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 7
stack: # TempProgram Function
string: ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID + $NUM ) ; }
while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 8
stack: # TempProgram TempFunction Block ) ParameterNum Parameter ( TempLeftParenthesis
string: ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID + $NUM ) ; }
while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 9
```

出错处理：

若语法分析过程中出现错误，则弹出错误窗口，并指出错误的步骤数。（压缩包中有 test_wrong_1.txt 和 test_wrong_2 的测试用例）

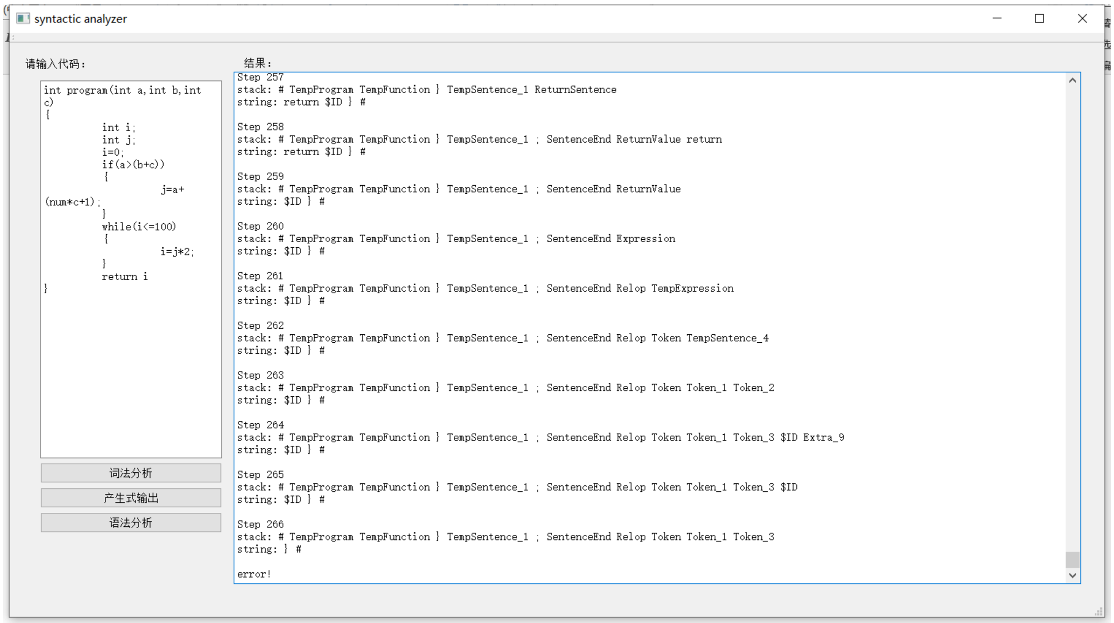


```
int program(int a,int b,int
c)
{
          int i;
          int j;
          i=0;
          if(a>(b+c))
          {
                    j=a+
(num*c+1);
          }
          while(i<=100)
          {
                    i=j*2;
          }
          return i
}
```

```
Step 1
stack: # Program
string: int $ID ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID +
$NUM ) ; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 2
stack: # SubProgram
string: int $ID ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID +
$NUM ) ; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 3
stack: # TempProgram Declaration
string: int $ID ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID +
$NUM ) ; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 4
stack: # TempProgram Type $ID int
string: int $ID ( int $ID , int $ID , int $                        ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID +
$NUM ) ; } while ( $ID <= $NUM ) { $ID = $I                        ; } #

Step 5
stack: # TempProgram Type $ID
string: $ID ( int $ID , int $ID , int $ID )                        ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID + $NUM )
; } while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 6
stack: # TempProgram Type
string: ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID + $NUM ) ; }
while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 7
stack: # TempProgram Function
string: ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID + $NUM ) ; }
while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 8
stack: # TempProgram TempFunction Block ) ParameterNum Parameter ( TempLeftParenthesis
string: ( int $ID , int $ID , int $ID ) { int $ID ; int $ID ; $ID = $NUM ; if ( $ID > ( $ID + $ID ) ) { $ID = $ID + ( $ID * $ID + $NUM ) ; }
while ( $ID <= $NUM ) { $ID = $ID * $NUM ; } return $ID ; } #

Step 9
```

Error: Step 266

通过右侧输出的结果，可以清楚看到是哪里出现了错误。



```
int program(int a,int b,int
c)
{
          int i;
          int j;
          i=0;
          if(a>(b+c))
          {
                    j=a+
(num*c+1);
          }
          while(i<=100)
          {
                    i=j*2;
          }
          return i
}
```

```
Step 257
stack: # TempProgram TempFunction } TempSentence_1 ReturnSentence
string: return $ID } #

Step 258
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd ReturnValue return
string: return $ID } #

Step 259
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd ReturnValue
string: $ID } #

Step 260
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd Expression
string: $ID } #

Step 261
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd Relop TempExpression
string: $ID } #

Step 262
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd Relop Token TempSentence_4
string: $ID } #

Step 263
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd Relop Token Token_1 Token_2
string: $ID } #

Step 264
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd Relop Token Token_1 Token_3 $ID Extra_9
string: $ID } #

Step 265
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd Relop Token Token_1 Token_3 $ID
string: $ID } #

Step 266
stack: # TempProgram TempFunction } TempSentence_1 ; SentenceEnd Relop Token Token_1 Token_3
string: } #

error!
```