总评成绩：_____

实验名称：实验 3.2 针对 paint 实验平台实现小型编译器

若申请当面验收，在后面横线上签上名字 _____

实验起评成绩：_____

姓名：__张**_____ 学号：__**********_班级：___计 135___

指定验收时间： ____第 17 周_____

电子文档： 提交时间 ___提前_____ 提前 / 按时 / 延时 / 未提交

格式正确____ 基本正确____ 不正确____

实验报告： 提交时间 ___提前_____ 提前 / 按时 / 延时

非常好____ 完 整____ 合 格____ 不合格____

验收记录：时间 _____ **顺序**_____

-
-
-
-
-
-
-
-
-
-
-
-

一、原创性声明

参考教材：编译原理、算法与数据结构。

原理以及实现由本人独立构思，独立实现

二、实验要求

1.实现一个编译器，将高级语言程序翻译为平台能够接受的指令序列，存入文件，在paint 平台下加载运行。

2.可以自己设计高级语言的词法和语法规则。

* 词法规则用正则表达式描述。
  包含对五类单词的定义：关键词、标识符、常量、界符、算符。
* 语法规则用文法描述，可以参考附录 A。
  TO 语言为最低要求。

三、完成情况

- **功能 1 ： 基本内容**

  制定出简单的编程语言，所制定语言能都完成基本的表达式功能，且能完成编译器前端的基本功能，即将高级语言编译为最终的目标语言，实现语言的基本编译功能。本实验中另外要求能够输出 paint 平台所能识别的基本语言，用于 paint 平台的指令输入。

  词法分析：规范程序使用词素规则，制定程序变量、关键字、标识符、常量等词素规范，完成基本的单词分离功能，并对分离出来的单词进行属性初始化，最终获得分离出来的词素，用于接下来的语法分析过程。

  语法分析：按照制定的编程语言，根据其语言特点制定语法规则，规则中不可含有二义性（如果有则手工消除），该过程完成语法的匹配，将每个词素作为节点，生成语法树（也可仅建立隐式语法树），完成基本语法分析功能。

  语法制导翻译：完成生成语法树的翻译，即遍历语法树，同时完成语言翻译，并相应的做出相应的规定动作，实现高级语言的最终翻译功能。

- **功能 2 ： 选做内容**

  在基本的 TO 语言基础上，增加较为复杂的语法规则，完善编程语言的语法规则，能支持更为复杂的语法功能，例如 IF 条件语句和 While 循环语句等。

- **功能 3 ： 自己扩展的内容**

  （1）较为完善的报错体系、警告提示功能。语法分析中添加变量声明、强制转换、比较表达式、浮点类型数据、符号数等较为复杂的语法功能。程序中设置的 paint 命令参数可为变量、正负号+变量或者是数字，增强编码灵活性（例：point －ID，-NUM）。

  （2）词法分析阶段：

  ①词素内加入行数这一属性，用于之后错误位置提示。词法分析中可完成对分离词素检验，不符合规范的予以提示。

  ②支持词法分析阶段的跳错编译。

  ③词素文本记录功能。

  （3）语法分析：文本记录语法分析阶段出入栈情况以及生成的语法树，且拥有较为全面的出错报错提示功能。

  （4）语法制导翻译：拥有完善的差错报错警告提示的功能，并对警告级别的提示做出规范化的自动纠错，并完成最终编译功能。

  （具体扩展功能详见创新和亮点）

四、实现方案

本编译器前端整体由三部分组成，分别为词法分析（scanner 类）、语法分析（parse 类）和语法制导翻译（trans 类），通过不同阶段函数调用实现编译器（**具体函数功能可见七、备注部分**）。具体实现如下：

词法分析：

主要完成单词流的分离，获得词素；且此阶段分离出的每个词素由四部分组成，分别为（索引项（index），字段类别（name），值（value），所在源文件行数（lineCoun）），其中 index 为语法分析时用于直接判别的终结符标志，字段类别记录下该词素的分类，其中分类共包含关键字（KEYWORD）、运算符（OP）、比较符号（COMP_SYM）、分号（SPL_SYM）、赋值号（ASS_SYM）、括号（BRA_SYM）、数字值（NUM）以及实例化的变量名称等。另外在该阶段完成了多个单词组成的写入命令的分离（如：set point size）。

具体分离时通过对输入代码“标准化”（trim 函数），即去掉输入中多余的空格回车等字符，并对一些字符进行分离（例如运算符、数字），从而获得标准的输入流，在进行单词分割和属性的初始化，并通过设置全局变量将分离出的词素返回主函数。

语法分析：

设置两个栈，分别存储字段和树节点，在出入栈的同时，通过之前做好的预测分析表作为驱动完成建树过程。具体方法为：①如果当前栈定为空串，直接弹出两个栈；

②如果当前栈为终结符（不为空串）,则与词素表中当前词素类型进行比对，相同则将其退出栈顶，否则则是出现匹配错误，直接报错。

③如果当前栈顶为非终结符，则将节点栈的栈顶作为节点、将预测分析表中对应的符号最为其子节点建立树，建树完成后两个栈分别弹出栈顶，且将建树时的子节点逆序压入节点栈中（字段符号栈同理）。

语义制导翻译：

语义制导翻译即为深度遍历树的过程，遍历的同时完成语义的分析，并触发相应的动作，从而完成语义的翻译过程。翻译的过程中完成对错误的提示和处理。

五、创新和亮点

（1）整体特点：较为复杂的文法设计,该编译器前段支持较为复杂的语法结构，除表达式外，增加两种条件语句（if、if…else…）、循环语句（while）、变量声明、强制转换、浮点数、整型位数控制、符号数等较为复杂语法结构，并支持几种语句的嵌套使用。文法较为复杂，灵活性强。

（2）词法分析:

①词法分析阶段出现词法错误时可以跳过错误继续编译，且输错错误位置（所在列数）、错误词素以及错误具体原因，出错原因分为使用关键字分为变量名称错误（变量只能有数字、字母、下划线，且以不能以数字开头）、使用关键字作为变量名称以及其他词法错误。

②支持代码注释功能，可以通过“//”进行注释，注释部分将在词法分析时自动跳过

③词法分析阶段分离出属性信息较为全面的词素，用于之后语法分析和报错警告提示信息。

④分离出的词素将会自动保存到文档中,用于纠错分析代码使用(保存路径:src//Token.txt)。

（3）语法分析：

①语法功能中添加类型自动强制转换功能。当赋值表达式左右数据类型不同时，自动数据类型转化，完成相应数据的存取，保证声明变量时类型属性；

②文件输出入语法分析的过程，即出入栈状态（保存路径：src//Stack.txt）,如果语法分析匹配成功时将会按照树层数输出树节点信息。

③出现语法匹配错误时输出详尽的的提示信息。错误信息有三种：终结符匹配错误、语法分析中间错误（提示当前错误信息后，将在控制台和 Tree.txt 中分别输出、输入中未匹配的词素）、程序结尾不完整错误（例如缺少 END 等错误，提示当前错误信息后，将在控制台和 Tree.txt 中分别输出、输入中栈中剩余未匹配的节点）。

（4）制导翻译：

①该阶段创建变量表，变量（全局变量）声明后加入该表，每次变量被调用或者数值发生改变时文件输出变量表,用于程序的调试和变量观察。

②设置严谨的警告和报错功能。制导翻译中共设置 3 中错误 4 种警告提示。设置如下：

A 错误类型：

i、运算时对浮点数取余错误

ii、变量未经声明即使用错误

iii、分母为 0 错误，

B 警告类型：

i、强制类型转换，出现在赋值语句(=)和比较表达式（>、<、==）中操作符左右的数据类型不相同时将会自动进行数据类型转换，并给与警告提示。

ii、整数越界，当出现在赋值号（=）右侧数值大于最大值时（默认 0x7fffffff），可能为程序中未注意的错误，将会给予警告提示。

iii、变量语句中变量重复声明警告，本编译器中全局变量重复定义无效

iv、变量未经初始化即使用警告。变量未初始化时默认为 0，且给予警告提示。

六、运行结果

给出尽可能完备的测试用例及测试结果(截图)

（1） 样例程序：

①测试程序（循环条件语句嵌套）：

```
BEGIN
  int id1,id2;
  float id3,id4;
  id1=20/(-5+-0.5)*2;//测试警告
  id2=id3;//测试警告
  id3=0.0;
  while(id2<12)//Repeat + IF测试
    id1=0;
    if(id2==3) id3=0.0; END
    if(id2==6) id3=0.0; END
    if(id2==9) id3=0.0; END
    while(id1<25)
       id3=id3+4.5;
       if(id2<3)//第一象限
          point id1,id3;
       else
         if(id2<6)//第二象限
            point -id1,id3;
         else
           if(id2<9)//第三象限
             point -id1,-id3;
           else //第四象限
             point id1,-id3;
           END
         END
       END
       id1=id1+5;//控制内层循环
    END
    id2=id2+1;//控制外层循环
   END
END
```

词法分析成功全部词素为：

(索引项	名字	值	所在源文件行)
(BEGIN	,KEYWORD	,BEGIN	,1)
(int	,KEYWORD	,int	,2)
(ID	,id1	,	,2)
(,	,SPL_SYM	,,	,2)
(ID	,id2	,	,2)
(;	,SPL_SYM	,;	,2)
(float	,KEYWORD	,float	,3)
(ID	,id3	,	,3)
(,	,SPL_SYM	,,	,3)
(ID	,id4	,	,3)
(;	,SPL_SYM	,;	,3)
(ID	,id1	,	,4)
(=	,ASS_SYM	,=	,4)
(NUM	,NUM	,20	,4)
(/	,OP	,/	,4)
((	,BRA_SYM	,(	,4)
(-	,OP	,-	,4)
(NUM	,NUM	,5	,4)
(+	,OP	,+	,4)
(-	,OP	,-	,4)
(NUM	,NUM	,0.5	,4)
()	,BRA_SYM	,)	,4)
(*	,OP	,*	,4)
(NUM	,NUM	,2	,4)
(;	,SPL_SYM	,;	,4)
(ID	,id2	,	,5)
(=	,ASS_SYM	,=	,5)
(ID	,id3	,	,5)
(;	,SPL_SYM	,;	,5)
(ID	,id3	,	,6)
(=	,ASS_SYM	,=	,6)
(NUM	,NUM	,0.0	,6)
(;	,SPL_SYM	,;	,6)
(while	,KEYWORD	,while	,7)
((	,BRA_SYM	,(	,7)
(ID	,id2	,	,7)
(<	,COMP_SYM	,<	,7)
(NUM	,NUM	,12	,7)
()	,BRA_SYM	,)	,7)
(ID	,id1	,	,8)
(=	,ASS_SYM	,=	,8)
(NUM	,NUM	,0	,8)
(;	,SPL_SYM	,;	,8)
(if	,KEYWORD	,if	,9)
((	,BRA_SYM	,(	,9)
(ID	,id2	,	,9)
(==	,COMP_SYM	,==	,9)

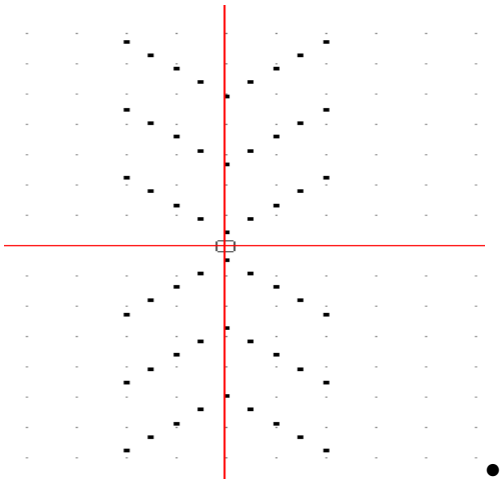
==========变量表==========

Name	value	type
id1	-7	int
id2		int
id3		float
id1	-7	int
id2	0	int
id3		float
id1	-7	int
id2	0	int
id3	0.0	float
id1	0	int
id2	0	int
id3	0.0	float
id1	0	int
id2	0	int
id3	4.5	float
id1	5	int
id2	0	int
id3	4.5	float
id1	5	int
id2	0	int
id3	9	float
id1	10	int
id2	0	int
id3	9	float
id1	10	int
id2	0	int
id3	13.5	float
id1	15	int

●源程序　　●词素表（部分）　　●变量更新表（部分）

```
==========语法分析结束，语义分析开始==========

编译中.....
【wearning】第4行 :赋值语句中=左右数据类型不相同，已强制类型转换
【wearning】第5行 变量id3未经初始化,编译器已将该变量的值默认为0，但其任未初始化
【wearning】第5行 :赋值语句中=左右数据类型不相同，已强制类型转换


==========编译成功==========
```

●控制台警告

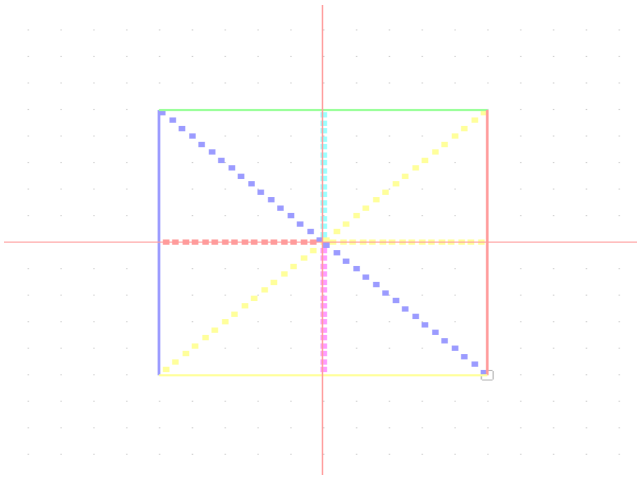


●运行结果表

②产生指令用于观察编译后效果（含 paint 运行）

```
BEGIN
   int a ;
   a=0;
   int b;
   b=50;
  reset;

  set point size 5;
  set color 1,0,0;
  while(a<b)   //x正半轴画点
     if(a%3==0)
       point a,0;
     END
     a=a+1;
  END

set color 1,1,0;
  a=0;
  while(a<b)  //x负半轴画点
     if(a%3==0)
       point -a,0;
     END
     a=a+1;
  END

set color 1,0,1;
  a=0;
  while(a<b)//y正半轴画点
     if(a%3==0)
       point 0,a;
     END
     a=a+1;
  END

set color 0,1,1;
  a=0;
  while(a<b)//y负半轴画点
     if(a%3==0)
       point 0,-a;
     END
     a=a+1;
  END

set color 1,1,0;
  a=0;
  while(a<b) //第一象限斜画点
     point a,a;
     a=a+3;
  END

set color 0,0,1;
  a=1;  //第二象限斜画点
  while (a<b)
    point -a,a;
    a=a+3;
  END

set color 1,1,0;
  a=1; //第三象限斜画点
  while (a<b)
    point -a,-a;
    a=a+3;
  END

set color 0,0,1;
  a=1; ///第四象限斜画点
  while (a<b)
    point a,-a;
    a=a+3;
  END

  set color 1,1,0;
  set line width 15;
        //环绕一周运动
  delay 1;
  rotate 90;
  delay 1;
  translate b,b;
  delay 1;
  translate 0,-50;
  delay 1;
  translate 0,-50;
  delay 1;
  translate -50,0;
  delay 1;
  translate -50,0;
  delay 1;
  translate 0,50;

  delay 1;
  translate 0,50;

  delay 1;
  translate 50,0;

  delay 1;
  translate 0,-50;
  delay 1;
  rotate 90;//旋转运动

  delay 1;
  scale 2.0,2.0;

  set color 0,0,1;
  delay 1;//下面画线操作
  line  50.0, 50, 50,-50;
  delay 1;
  set color 0,1,0;
  line  50,-50.00,-50,-50;
  delay 1;
  set color 1,0,0;
  line -50,-50,-50.000, 50;
  delay 1;
  set color 1,1,0;
  line -50, 50, 50, 50.0000

  scale ;
  move to 50,50;//移动操作
  delay 1;
  move to 50,-50;
  delay 1;
  move to -50,-50;
  delay 1;
  move to -50,50;

END
```

●源代码



●paint 运行结果

```
词法分析成功全部词素为：
(索引项        名字        值        所在源文件行)
(BEGIN         ,KEYWORD    ,BEGIN     ,1)
(int           ,KEYWORD    ,int       ,2)
(ID            ,a          ,          ,2)
(;             ,SPL_SYM    ,;         ,2)
(ID            ,a          ,          ,3)
(=             ,ASS_SYM    ,=         ,3)
(NUM           ,NUM        ,0         ,3)
(;             ,SPL_SYM    ,;         ,3)
(int           ,KEYWORD    ,int       ,4)
(ID            ,b          ,          ,4)
(;             ,SPL_SYM    ,;         ,4)
(ID            ,b          ,          ,5)
(=             ,ASS_SYM    ,=         ,5)
(NUM           ,NUM        ,50        ,5)
(;             ,SPL_SYM    ,;         ,5)
(reset         ,KEYWORD    ,reset     ,6)
(;             ,SPL_SYM    ,;         ,6)
(set point size,KEYWORD    ,set point size,8)
(NUM           ,NUM        ,5         ,8)
(;             ,SPL_SYM    ,;         ,8)
(set color     ,KEYWORD    ,set color ,9)
(NUM           ,NUM        ,1         ,9)
(,             ,SPL_SYM    ,,         ,9)
(NUM           ,NUM        ,0         ,9)
(,             ,SPL_SYM    ,,         ,9)
(NUM           ,NUM        ,0         ,9)
(;             ,SPL_SYM    ,;         ,9)
(while         ,KEYWORD    ,while     ,10)
((             ,BRA_SYM    ,(         ,10)
(ID            ,a          ,          ,10)
(<             ,COMP_SYM   ,<         ,10)
(ID            ,b          ,          ,10)
()             ,BRA_SYM    ,)         ,10)
(if            ,KEYWORD    ,if        ,11)
((             ,BRA_SYM    ,(         ,11)
(ID            ,a          ,          ,11)
(%             ,OP         ,%         ,11)
(NUM           ,NUM        ,3         ,11)
(==            ,COMP_SYM   ,==        ,11)
(NUM           ,NUM        ,0         ,11)
()             ,BRA_SYM    ,)         ,11)
(point         ,KEYWORD    ,point     ,12)
(ID            ,a          ,          ,12)
(,             ,SPL_SYM    ,,         ,12)
(NUM           ,NUM        ,0         ,12)
(;             ,SPL_SYM    ,;         ,12)
(END           ,KEYWORD    ,END       ,13)
```

●词素表

```
字符栈入栈pragram

当前栈为：program
字符出栈program
字符栈入栈program2
字符栈入栈BEGIN

当前栈为：BEGIN program2
字符出栈BEGIN

当前栈为：program2
字符出栈program2
字符栈入栈END
字符栈入栈Stmt-List

当前栈为：Stmt-List END
字符出栈Stmt-List
字符栈入栈Stmt-List2
字符栈入栈Stmt

当前栈为：Stmt Stmt-List2 END
字符出栈Stmt
字符栈入栈Declare-Stmt

当前栈为：Declare-Stmt Stmt-List2 END
字符出栈Declare-Stmt
字符栈入栈;
字符栈入栈ID-List
字符栈入栈int

当前栈为：int ID-List ; Stmt-List2 END
字符出栈int

当前栈为：ID-List ; Stmt-List2 END
字符出栈ID-List
字符栈入栈ID-List2
字符栈入栈ID

当前栈为：ID ID-List2 ; Stmt-List2 END
字符出栈ID

当前栈为：ID-List2 ; Stmt-List2 END
字符出栈ID-List2

当前栈为：@ ; Stmt-List2 END

当前栈为：; Stmt-List2 END
字符出栈;
```

●出入栈表

```
==================变量表====================
Name value type

a     0     int
=============================================

a     0     int
b     50     int
=============================================

a     1     int
b     50     int
=============================================

a     2     int
b     50     int
=============================================

a     3     int
b     50     int
=============================================

a     4     int
b     50     int
=============================================

a     5     int
b     50     int
=============================================

a     6     int
b     50     int
=============================================

a     7     int
b     50     int
=============================================

a     8     int
b     50     int
=============================================

a     9     int
b     50     int
=============================================

a     10     int
b     50     int
=============================================
```

●变量表

（1）词法分析：

①错误程序代码：

```
BEGIN
int 1a ;  //变量不能以数字开头
float x;
   x=5*-1a*1.5+2*8;
set window sizel; //不能用关键字作为变量名称
int a;
a=3+10;
file "C:\Program Files";
a=6*3;
file "C:  ; //路径缺少右
int .b ;
int ab&hd;
END
```

●源程序

```
词法分析第2行定义不符合规范！！变量不能以数字开头，出错词素：1a
词法分析第4行定义不符合规范！！变量不能以数字开头，出错词素：1a
词法分析第5行定义不符合规范！！！不能用关键字作为变量，出错词素：set window size
词法分析第10行定义不符合规范！！！""内路径缺少右"
词法分析第11行定义不符合规范！！变量不能以数字开头，出错词素：.b
词法分析第11行定义不符合规范！！！变量不能以数字开头，出错词素：.b
词法分析第12行定义不符合规范！！！出错词素：ab&hd
词法分析成功全部词素为：
(索引项        名字          值       所在源文件行)
(BEGIN         ,KEYWORD    ,BEGIN      ,1)
(int           ,KEYWORD    ,int        ,2)
(NUM           ,NUM        ,1a         ,2)
(;             ,SPL_SYM    ,;          ,2)
(float         ,KEYWORD    ,float      ,3)
(ID            ,x          ,           ,3)
(;             ,SPL_SYM    ,;          ,3)
(ID            ,x          ,           ,4)
(=             ,ASS_SYM    ,=          ,4)
```

●编译错误结果编译结果（控制台和 src//Token.txt 中）

（2） 语法分析：

①程序结尾赘余词素导致语法错误

```
BEGIN
   int a ;
   a=1;
   float x;
   x=5*-a*1.5+2*8;
   point a,x;

point a,x;
///此处缺少结束END关键字
```

●源程序

●错误信息（控制台和 src//Tree.txt 中））

```
【error】语法错误，出现不被期望字符，错误位置第8行,匹配错误的非终结符:Stmt-List2

匹配完成时，栈中还有字符，语法错误
 栈中剩余节点内容为：
(Stmt-List2,Stmt-List2,Stmt-List2,-1)
(END,END,END,-1)


语法分析发现错误！！！语义分析未进行，请调试.....
```

②程序中间非终结符匹配错误

```
BEGIN
   int a ;
   a=1;
   float x;
   x=5*-a*1.5+2*8;
   point a,x;
END
point a,x;
END
```

```
匹配出错后出现剩余未匹配词素，且剩余词素为 ：
(point,KEYWORD,point,8)
(ID,a,,8)
(,,SPL_SYM,,,8)
(ID,x,,8)
(;,SPL_SYM,;,8)
(END,KEYWORD,END,9)


语法分析发现错误！！！ 语义分析未进行，请调试......
```

●源程序　　●错误信息（控制台和 src//Tree.txt 中

③终结符匹配错误

```
BEGIN
  if x(true)
     point 2,3;
END
```

```
【error】第2行的中间字符ID匹配错误期望的字符为：<

语法分析发现错误！！！ 语义分析未进行，请调试......
```

●源程序　　●错误信息(控制台)

（3）语义分析警告及错误：

①基本警告错误信息信息

```
BEGIN
  int a,c;
  float b;
  a=111111111111111;
  a=-2*2.4+-5;
  a = c;
  a=2.0;
  if(a>0.5)
    b=a;
    x=x+11;
   END
END
```

●源程序

```
=====================语法分析结束，语义分析开始============================

编译中.....
【wearning】第4行 :等式右边数值超过整型的最大值
【wearning】第5行 :赋值语句中=左右数据类型不相同，已强制类型转换
【wearning】第6行 变量c未经初始化,编译器已将该变量的值默认为0，但其任未初始化
【wearning】第7行 :赋值语句中=左右数据类型不相同，已强制类型转换
【wearning】第8行 :条件语句中条件比较时>号左右比较值类型不同,已强制类型转换！！
【wearning】第9行 :赋值语句中=左右数据类型不相同，已强制类型转换
【error】第10行 变量x未经声明即使用，非法赋值！！！


=====================================编译失败==================================
```

●错误信息（控制台）

②运算错误

```
BEGIN
  int a;
  float b;
  b=2.3;
  if(true)
    a=b%2;
  END
END
```

```
=====================语法分析结束，语义分析开始=============================

编译中.....
【error】第6行语义翻译出现浮点数取余


=====================================编译失败=================================
```

●源程序　　●错误信息（控制台）

七、备注

1、具体函数功能一览表

类	函数名称	函数功能
词法分析(scanner)	bool isOperator(char ch)	判断是否为运算符
	void outtoken(token alltoken[],int tokenNum)	向src//Token.txt写入分离出来的Token
	string lineTrim(string str,int lineCoun)	对源程序规格化，去除空格等无用字符，并对串特殊化处理，用于之后单词分离
	bool isKeyword(string word)	判断是否为关键字
	bool isIntNum(string word,int lineCoun)	判断是否为整型数字
	bool isFloNum(string word,int lineCoun)	判断是否为浮点型数字
	bool isID(string word)	判断是否符合变量命名要求
	bool scannerFuntion(string line,int lineCoun,token alltoken[],int &tokenNum)	获得词素函数（词法分析主函数）
语法分析(parser)	int findTerminal(token ttok)	查找预测分析表中终结符行数
	int findNTerminal(token ttok)	查找预测分析表中非终结符行数
	void printStack(stack <token> ssta)	语法分析同时输出入栈出栈情况（（控制台和文件(路径src//Tree.txt））
	void gettoken(string gramm,token splitToken[],int &tokNum)	获得预测分析表中关键字
	void buildTree(node *father,stack <node *> &tempsta)	生成语法树
	void pushstack(string gramm,stack <token> &sta,stack <node *> &nodeSta,node *father)	入栈出栈处理
	void printTree (node *p)	输出生成的树（控制台和文件（路径src//Tree.txt））
	bool parse::function(token alltoken[],int tokenNum,stack <node * > &nodeSta,stack	完成语法分析（语法分析主函数）
语义分析（trans）	void printResult(bool b)	输出编译结果
	int findIDIndex(string str)	查找变量str所在的变量表下标位置
	string judgeIntOrFloat(string num)	判断是否为整数或者是浮点数
	string getFloString(float flo)	将float数转化为字符串
	bool trans getStandarExp(node * root)	翻译Expr或者Term表达式
	void trans printID()	文件输出变量表（每次变量别使用或者赋值时调用）
	void getCompExp(node *&root)	翻译Expr Comp-OP Expr类型比较表达式
	void transFunction(node *&root)	完成语义分析（语义制导翻译主函数）

2、预测分析表（部分）

3、平台 BUG

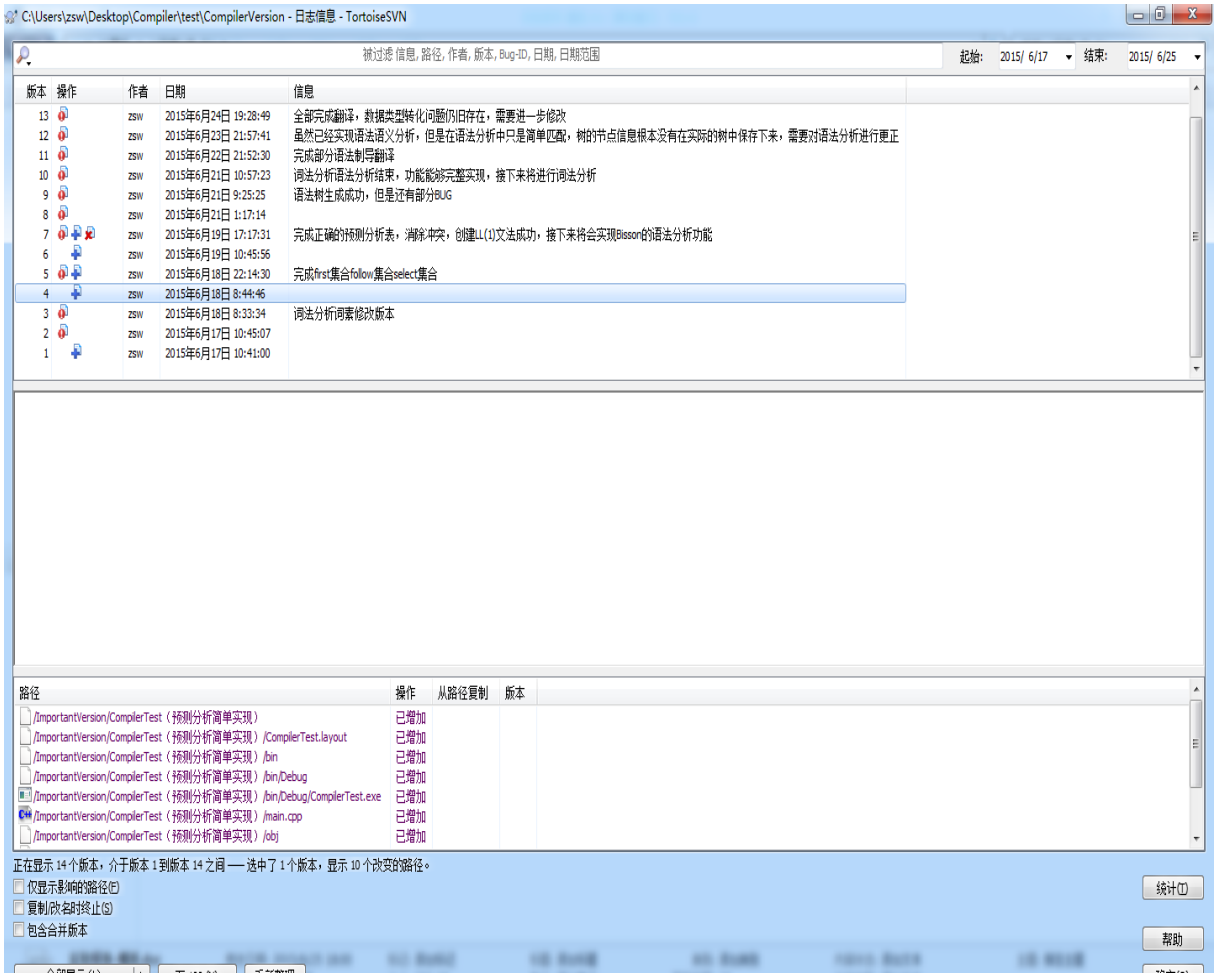
①Line 指令有时候无法画出线（只在一个象限中时），例如 line 10,20,30,40；只能通过将四个参数中某一个参数转化为浮点型才能画出。

②无法从文件中读取完成 file (:filename)?指令。

4、完善

词法分析时完成了跳过错误后继续编译的过程，预期语法分析时也可以对栈进行操作，实现跳错编译，但是由于时间等原因未能完整实现，需进一步完善。

5、版本记录（开始部分未保存）



（更多详细测试和介绍见电子版）

八、程序代码部分

（1）头文件部分：

①自己写的数据结构类型

```
#ifndef MYCLASS_H
#define MYCLASS_H

#include <iostream>
using namespace std;
struct token
{
    token(string ind, string nam, string val, int l){ ///构造函数
        index = ind;
        name = nam;
        value = val;
        loca = l;
    }
    token(){}    ///构造函数
    string index;
    string name ;
    string value;
    int loca;
};

#define Maxsize 10000

typedef struct Node{
  token tokenn;
  Node * child[10];
  int childNum;
}node;

struct ID{
string name;
string intOrFlo;
string value ;
};

struct IDTable{ ///变量表
    IDTable(){idnum=0;}
    ID id[Maxsize];
    int idnum;
};

static string preTable[44][40]={   ///考虑到编码问题，由于 ε 编译过程中会出现乱码，因此不使用 ε 表示空串，而是用@表示空串
      {"BEGIN    program2","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","",""},///Program
```

{"","END","Stmt-List END","","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END" ,"","","","","","","","","","","","","","","","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END","Stmt-List END"},///program2

{"","", "Stmt Stmt-List2","","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","","","","","","","","","","","","","","","","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2"},///Stmt-List

{"","@","Stmt Stmt-List2","@","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","","","","","","","","","","","","","","","","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2","Stmt Stmt-List2"},///Stmt-List2

{"","","If-Stmt","","Repeat-Stmt","Declare-Stmt","Declare-Stmt","Assign-Stmt","","","","","","","","","","","","","","","","","Command","Command"," Command" ,"Command" ,"Command" ,"Command" ,"Command" ,"Command" ,"Command" ,"Command" ,"Command" ,"Command" ,"Command" ,"Command" ,"Command"},///Stmt

{"","","","","","","","ID = Expr ;",""},///Assign-Stmt

{"","","if (Compare-Exp) Stmt-List If-Stmt2","" },///If-Stmt

{"","END","","else Stmt-List END",""},///If-stmt2

{"","","","","","int ID-List ;","float ID-List ;",""},///Declare-Stmt

{"","","","","","","","ID ID-List2",""},///ID-List

{"","","","","","","","","","","","","","","","","","","@",", ID ID-List2","","","","","","","","","","","","","","","","","","",""},///ID-List2

{"","","","","while (Compare-Exp) Stmt-List END",""},///Repeat-Stmt

{"","","","","","","","Expr Comp-OP Expr","Expr Comp-OP Expr","true","false","","Expr Comp-OP Expr","","","","Expr Comp-OP Expr",""},///Compare-Exp

{"","Help-Com ;","File-Com ;","Delay-Com ;","Reset-Com ;","Point-Com ;","Line-Com ;","Rotate-Com ;","Translate-Com ;","Scale-Com ;","Get-windsiz-Com ;","Set-Window-Com ;","Set-Color-Com ;","Set-PointSiz-Com ;","Set-LineWid-Com ;","MoveTo-Com ;","LineTo-Com ;"},///Command

{"","","","","","","","Term Expr2","Term Expr2","","","","Term Expr2","","","","Term Expr2",""},///Expr

{"","","","","","","","","","","","Add-OP Term Expr2","Add-OP Term Expr2","","","","","@","@","","@","@","@","","","","","","","","","","","","","","","","",""},///Expr2

{"","","","","","","","Factor Term2","Factor Term2","","","","Factor Term2","","","","Factor Term2",""},///Term

{"","","","","","","","","","","","","@","@","Multi-OP Factor Term2","Multi-OP Factor Term2","Multi-OP Factor Term2","","@","@","","@","@","@","","","","","","","","","","","","","","","","",""},///Term2

{"","","","","","","","ID-NUM","ID-NUM","","","","ID-NUM","","","","(Expr)",""},/// Factor

{"","","","","","","","ID","NUM","","","", "- ID-NUM2""",""},///ID-NUM

{"","","","","","","","ID","NUM",""},///ID-NUM2

```
{"","","","","","","","","","","","+","-","","","","","","","","","","","","","","","","","","","","","","","","","","",""},///ADD-OP
{"","","","","","","","","","","","","*","/","%","","","","","","","","","","","","","","","","","","","","","","","","",""},///Multi-OP
{"","","","","","","","","","","","","","","","","","","","",">","<","==","","","","","","","","","","","","","","","","",""},///Comp-OP
{"","","","","","","","","","","","","","","","","","","","","","","","","file    File-Com2","","","","","","","","","","","","","","",""},///File-Com
{"","","","","","","","","","","","","","","","","","","@","","","","","filename","","","","","","","","","","","","","","","",""},///File-Com2
{"","","","","","","","","","","","","","","","","","","","","","","","","","delay    Param-1","","","","","","","","","","","","","",""},///Delay-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","set window size Param-1","","","","",""},///Set-Window-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","set color    Param-3","","","",""},///Set-Color-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","set point size Param-1","","",""},///Set-PointSiz-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","set line width Param-1","",""},///Set-LineWid-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","point    Param-2","","","","","","","","","","","",""},///Point-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","scale    Param-2","","","","","","","",""},///Scale-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","translate    Param-2","","","","","","","","",""},///Translate-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","reset","","","","","","","","","","","","",""},///Reset-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","get window size","","","","","",""},///Get-windsiz-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","move to ID-NUM , ID-NUM",""},///MoveTo-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","line to ID-NUM , ID-NUM"},///LineTo-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","line ID-NUM , ID-NUM , ID-NUM , ID-NUM","","","","","","","","","",""},///Line-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","rotate    ID-NUM","","","","","","","","",""},///Rotate-Com
{"","","","","","","","","","","","","","","","","","","","","","","","","help","","","","","","","","","","","","","","",""},///Help-Com
{"","","","","","","","ID-NUM","ID-NUM","","","","ID-NUM","","","","","","@","","","","","","","","","","","","","","","","","","","","",""},///Param-1
{"","","","","","","","ID-NUM , ID-NUM","ID-NUM , ID-NUM","","","","ID-NUM , ID-NUM","","","","","","@","","","","","","","","","","","","","","","","","","","","",""},///Param-2
{"","","","","","","","ID-NUM , ID-NUM , ID-NUM","ID-NUM , ID-NUM , ID-NUM","","","","ID-NUM , ID-NUM , ID-NUM","","","","","","@","","","","","","","","","","","","","","","","","","","","",""}///Param-3
};
```

②词法分析头文件

```
#ifndef SCANNER_H
#define SCANNER_H
#include "myclass.h"
#include <iostream>
#include <cstdio>
#include <fstream>
using namespace std;

class scanner
{
	private:
```

```
    public:
        scanner();
        virtual ~scanner();
            bool    isOperator(char ch);
        string    lineTrim(string str,int lineCoun);
        bool isKeyword(string word);
        void outtoken(token alltoken[],int tokenNum);
        bool isIntNum(string word,int lineCoun);
        bool isFloNum(string word,int lineCoun);
        bool isID(string word);
        bool scannerFuntion(string line,int lineCoun,token alltoken[],int &tokenNum);
    protected:
};

#endif // SCANNER_H
```

③语法分析头文件

```
#ifndef PARSE_H
#define PARSE_H
#include "myclass.h"
#include <iostream>
#include <stack>

class parse
{
    public:
        parse();
        virtual ~parse();
          void    printTree (node *p);
        bool    function(token alltoken[],int tokenNum,stack <node*> &nodeSta,stack <token> sta,node * &root);
        void printStack(stack <token> ssta);
    protected:
    private:
};

#endif // PARSE_H
```

④语义翻译头文件

```
#ifndef TRANS_H
#define TRANS_H
#include "myclass.h"

class trans
{
    public:
```

```
        trans();
        virtual ~trans();
        void     printID();
        void    printResult(bool b);
        void    transFunction(node *&root);
        bool    getStandarExp(node * root);
        string getFloString(float flo);
    private:
};

#endif // TRANS_H
```

（3） 函数部分

①主函数：

```
#include <iostream>
#include <iomanip>
#include <cstdio>
#include "myclass.h"
#include "scanner.h"
#include "parse.h"
#include "trans.h"
#include <fstream>
#include <stack>
using namespace std;

int main()
{
   token alltoken[Maxsize];
   int tokenNum=0;
   ifstream in("source/src.txt");
    string temp;
    int lineCoun=1;
    scanner myscanner;
    bool lexmark=1;
    while( getline(in,temp)    )
    {
       if(!myscanner.scannerFuntion(temp,lineCoun,alltoken,tokenNum))
            lexmark=0;
       lineCoun++;
    }
    myscanner.outtoken(alltoken,tokenNum); ///输出分离出的词素
   if(lexmark==0) {
     cout<<"======================词法分析出现错误，编译失败结束!!! ====================="<<endl;
     return 0;
    }
```

```
    cout<<"================词法分析结束，语法分析开始=======================\n";
    stack <token> sta;
    stack <node *> nodeSta;
    node *root=NULL;
    parse myParse;
    bool mark = myParse.function(alltoken,tokenNum,nodeSta,sta,root); ///语法分析开始
    if(!mark) { ///语法分析失败
            cout<<endl<<endl<<"语法分析发现错误!!! 语义分析未进行，请调试......"<<endl;
            return 0;
    }
     cout<<"=======================语法分析结束，语义分析开始==============================\n 编译中.....\n";

     trans tra;
     tra.transFunction(root);///语义翻译
     tra.printID();
     tra.printResult(1); ///输出语义翻译结果
    return 0;
}
```

②词法分析函数

```
#include "scanner.h"
#include "myclass.h"
#include <iostream>
#include <cstdio>
#include <iomanip>
#include<stdlib.h>
#include <fstream>
using namespace std;
ofstream    outToken("source/Token.txt",ios::out);///将词法分析的分离出的词素写入 Token.txt 中
bool mark=1;
int keywordNum = 27;
string keyword[] = {"help","file","delay","reset","set window size",
                    "get window size","set color",
                    "set point size","set line width",
                    "move to","line to","point","line","rotate",
                    "ratate","scale","translate","if","then",
                    "else","BEGIN","END","while","int","float",
                    "true","false"
                    };
bool scanner::isOperator(char ch){
    if(ch=='+'||ch=='-'||ch=='*'||ch=='/'||ch=='%')
            return true;
    return false;
```

```
}

void scanner::outttoken(token alltoken[],int tokenNum){
    outToken<<"词法分析成功全部词素为：\n"<<"(索引项              名字              值          所在源文件行)"<<endl;
    printf("词法分析全部词素为：\n((%-14s,%-10s,%-5s,%s)\n","索引项","名字","值","所在源文件行");


    for(int i=0;i<tokenNum;i++){
       printf("(%-14s,%-10s,%-5s,%d)\n",alltoken[i].index.c_str(),alltoken[i].name.c_str(),alltoken[i].value.c_str(),alltoken[i].loca );

outToken<<setiosflags(ios::fixed)<<setiosflags(ios::left)<<"("<<setw(14)<<alltoken[i].index<<setw(1)<<","<<setw(10)<<alltoken[i].name<<setw(1)
<<","<<setw(10)<<alltoken[i].value<<setw(1)<<','<<alltoken[i].loca<<")"<<endl;
    }
}
string scanner::lineTrim(string str,int lineCoun)///调整为标准模式
{
     int len = str.length();
     string temp="";
     for(int i=0; i<len; i++)
     {
       if(      (temp.length()==0&&str[i]==' ')  ||  (  temp.length()>0&&(temp[temp.length()-1]==' '||temp[temp.length()-1]=='\t')&&(str[i]==' '||str[i]=='\t') )    )///去除空格或制表符
            continue;
        else if(str[i]==';'){
          if((temp.length()!=0 && temp[temp.length()-1]!=' '))
              temp+=" ";
          temp=temp+str[i]+" ";
        }
        else if(str[i]=='\"'){
          string ttemp="";
           if((temp.length()!=0 && temp[temp.length()-1]!=' '))
              ttemp+=" ";
          ttemp+="\"";
          i++;
          while(i<len&&str[i]!='\"'){
            if(str[i]==' '||str[i]=='\t')
              {
                  i++;
                  continue;
              }
              ttemp+=str[i++];
          }
          if(i<len&&str[i]=='"'){
              ttemp=ttemp+str[i++]+" ";
              temp+=ttemp;
```

```
            }

            else{
                printf("词法分析第%d 行定义不符合规范!!! \"\"内路径缺少右\"\n",lineCoun);
                outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!!! \"\"内路径缺少右\""<<endl;
                mark=0;
                while(str[i]!=' ')
                  i++;
            }
        }
        else if(str[i]=='/'    ){ ///  为注释
            if(i+1<len&&str[i+1]=='/')
                return temp;
            else{    ///分离为除号
                if((temp.length()!=0 && temp[temp.length()-1]!=' '))
                temp+=" ";
                temp=temp+str[i]+" ";
            }
        }
        else if( isOperator(str[i]) ||    str[i]=='(' || str[i]==')' || str[i]=='<'    || str[i]=='>' ||str[i]==',' ){          ///运算符和 (     )   >     <
            if((temp.length()!=0 && temp[temp.length()-1]!=' '))
                temp+=" ";
            temp=temp+str[i]+" ";
        }
        else if( str[i]=='=' ){            /// = ==
            if(temp.length()==0||(temp.length()!=0 && temp[temp.length()-1]!=' '))
                temp+=" ";
            if(i+1<len&&str[i+1]=='=')
                temp=temp+"=="+" ",i++;
            else
                temp=temp+str[i]+" ";
        }
        else{
            temp+=str[i];
        }
    }
    return temp;
}

bool scanner::isKeyword(string word){
    for(int i=0;i<keywordNum;i++)
        if(word==keyword[i])
            return true;
    return false;
}
```

```
bool scanner::isIntNum(string word,int lineCoun){
    int len = word.length();
  for(int i=0;i<len;i++)
    if(!(word[i]<='9'&&word[i]>='0'))
        if(i!=0&&word[i]!='.')
        {
            printf("词法分析第%d 行定义不符合规范!!! 变量不能以数字开头，出错词素：",lineCoun);
            outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!! 变量不能以数字开头，出错词素："<<word<<endl;
            cout<<word<<endl;
            mark=0;
        }
        else
            return false;
    return true;
}

bool scanner::isFloNum(string word,int lineCoun)
{
    int len = word.length();
    bool mark=0;
    for(int i=0; i<len; i++)
        if(word[i]<='9'&&word[i]>='0')
            continue;
        else if(   mark==0&&word[i]=='.'   )
            mark=1;
        else if(i!=0){
            printf("词法分析第%d 行定义不符合规范!!! 变量不能以数字开头，出错词素：",lineCoun);
            outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!! 变量不能以数字开头，出错词素："<<word<<endl;
            cout<<word<<endl;
           mark=0;
        }
        else
            return false;
    if(word[0]=='.'||word[len-1]=='.')
    {
        printf("词法分析第%d 行定义不符合规范!!! 数字中不能有字符，出错词素：",lineCoun);
        outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!!! 变量不能以数字开头，出错词素："<<word<<endl;
        cout<<word<<endl;
        mark=0;
    }
    return true;
}
bool scanner::isID(string word){ ///判断是否符合变量规则
  int len = word.length();
```

```
    for(int i=0;i<len;i++)
        if( (i==0&&((word[i]<='z'&&word[i]>='a') ||    (word[i]<='Z'&&word[i]>='A') ||    word[i]=='_'))
            ||(i!=0&&(((word[i]<='z'&&word[i]>='a') ||    (word[i]<='Z'&&word[i]>='A') || (word[i]<='9'&&word[i]>='0') || word[i]=='_')) ) )
              continue;
        else
          return false;
return true;
}
bool scanner::scannerFuntion(string line,int lineCoun,token alltoken[],int &tokenNum){
    line = lineTrim( line,lineCoun );
    if(line=="\n"||line==" "||line=="\t"||line=="") ///空白行不做处理
         return 1;
    /// cout<<line<<endl;    ///测试
    int i=0,len = line.length();

    while(i<len){

        string word="";
        while( line[i]!=' '&&line[i]!='\t' && i<len )
           word+=line[i++];

        if(word=="set"){///获得包含 set 字符的指令串
              int j=i+11;
              for(;i<=j&&i<len;i++){
                    word+=line[i];
                  if(i==j-6&&word=="set color"){
                        i++;
                        break;
                  }
                  if(i==j-1&&word=="set point size"){
                        i++;
                        break;
                  }
                  if(i==j-1&&word=="set line width"){
                        i++;
                        break;
                  }
              }
              if(!isKeyword(word) ||(    i<len && line[i]!='\t'&& line[i]!=' '    ) ){
                 printf("词法分析第%d 行定义不符合规范!!! 不能用关键字作为变量，出错词素：",lineCoun);
                 outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!!! 不能用关键字作为变量，出错词素："<<word<<endl;
                     cout<<word<<endl;
                 mark=0;
                 while(line[i]!=' ')
                     i++;
```

```
        }
        else{
            alltoken[tokenNum].name="KEYWORD";
            alltoken[tokenNum].index = word;
            alltoken[tokenNum].loca=lineCoun;
            alltoken[tokenNum++].value=word;
        }
}
else if(word=="line"){///获得包含 line 字符的指令串;
    if(i+2<len&&word+line[i]+line[i+1]+line[i+2]=="line to" ){ ///当前字符为 line to
        if( i+3>=len ||(line[i+3]=='\t'||line[i+3]==' ') ){    ///保证 line to  后面紧跟的没有其他字母，有的话说明不是 line to,使用了关键字报错
          word=word+line[i]+line[i+1]+line[i+2];
          i+=3;
          alltoken[tokenNum].name="KEYWORD";
          alltoken[tokenNum].index = word;
          alltoken[tokenNum].loca=lineCoun;
          alltoken[tokenNum++].value=word;
        }
        else{
            cout<<(i+3>=len)<<(line[i+3]=='\t')<<(line[i+3]==' ')<<endl;
            printf("a 词法分析第%d 行定义不符合规范!!! \n 不能用关键字作为变量，出错词素：",lineCoun);
            outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!!! 不能用关键字作为变量，出错词素："<<word<<endl;
            cout<<word<<endl;
            mark=0;
            while(line[i]!=' ')
              i++;
        }
    }
    else{
        alltoken[tokenNum].name="KEYWORD";
        alltoken[tokenNum].index = word;
        alltoken[tokenNum].loca=lineCoun;
        alltoken[tokenNum++].value=word;
    }

}
else if(word=="get"){    ///含 get 关键字分离
    int j=i+11;
    for(;i<=j&&i<len;i++)
        word+=line[i];
    if(!isKeyword(word) ||(    i<len && line[i]!='\t'&& line[i]!=' '    ) ){
        printf("词法分析第%d 行定义不符合规范!!! \n 不能用关键字作为变量，出错词素：",lineCoun);
        outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!!! 不能用关键字作为变量，出错词素："<<word<<endl;
        cout<<word<<endl;
        mark=0;
```

```
            while(line[i]!=' ')
                i++;
            }
            else{
              alltoken[tokenNum].name="KEYWORD";
              alltoken[tokenNum].index = word;
              alltoken[tokenNum].loca=lineCoun;
              alltoken[tokenNum++].value=word;
            }
}
else if(word=="move"){
     int j=i+2;
      for(;i<=j&&i<len;i++)
            word+=line[i];
      if(!isKeyword(word) ||(    i<len && line[i]!='\t'&& line[i]!=' '    ) ){
          printf("词法分析第%d 行定义不符合规范!!! \n 不能用关键字作为变量，出错词素：",lineCoun);
          outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!!! 不能用关键字作为变量，出错词素："<<word<<endl;
          cout<<word<<endl;
          mark=0;
          while(line[i]!=' ')
              i++;
      }
      else{
          alltoken[tokenNum].name="KEYWORD";
          alltoken[tokenNum].index = word;
          alltoken[tokenNum].loca=lineCoun;
          alltoken[tokenNum++].value=word;
      }
}

else if(isKeyword(word)){    ///关键字
      alltoken[tokenNum].name="KEYWORD";
      alltoken[tokenNum].index = word;
      alltoken[tokenNum].loca=lineCoun;
            alltoken[tokenNum++].value = word;
}
else if(isIntNum(word,lineCoun)){
      alltoken[tokenNum].name= "NUM";    ///整型数字
      alltoken[tokenNum].index = "NUM";
      alltoken[tokenNum].loca=lineCoun;
      alltoken[tokenNum++].value = word;
      }
else if(isFloNum(word,lineCoun)){
      alltoken[tokenNum].name= "NUM";     ///浮点型数字
      alltoken[tokenNum].index = "NUM";
```

```
        alltoken[tokenNum].loca=lineCoun;
        alltoken[tokenNum++].value = word;
        }
    else if(isID(word)){          ///变量
        alltoken[tokenNum].name=word;
        alltoken[tokenNum].loca=lineCoun;
        alltoken[tokenNum].index = "ID";
        alltoken[tokenNum++].value ="";
    }
    else if(word=="+"||word=="-"||word=="*"||word=="/"||word=="%") {    ///运算符
        alltoken[tokenNum].name="OP";
        alltoken[tokenNum].loca=lineCoun;
        alltoken[tokenNum].value = word;
        alltoken[tokenNum++].index = word;

        }
    else if(word==">"||word=="<" || word=="=="){      ///比较符号
        alltoken[tokenNum].name="COMP_SYM";
        alltoken[tokenNum].loca=lineCoun;
        alltoken[tokenNum].index = word ;
        alltoken[tokenNum++].value = word;

        }
    else if(word[0]==\""){    ///获取文件名
          while(i<len&&line[i]!=' ')
              word+=line[i++];
          word.erase(0,1);
          word.erase(word.length()-1,1);
          alltoken[tokenNum].value = word;
          alltoken[tokenNum].name = "filename";
          alltoken[tokenNum].loca=lineCoun;
        alltoken[tokenNum++].index = "filename";
    }
    else if(word=="="){       ///赋值符号
        alltoken[tokenNum].name="ASS_SYM";
        alltoken[tokenNum].index = word;
        alltoken[tokenNum].loca=lineCoun;
        alltoken[tokenNum++].value = word;
    }
    else if(word=="("||word==")"){
        alltoken[tokenNum].name="BRA_SYM";
        alltoken[tokenNum].loca=lineCoun;
        alltoken[tokenNum].index = word;
        alltoken[tokenNum++].value = word;
    }
```

```
        else if(word==";"|| word == ","){
            alltoken[tokenNum].name="SPL_SYM";
            alltoken[tokenNum].index = word;
            alltoken[tokenNum].loca=lineCoun;
            alltoken[tokenNum++].value = word;
        }
        else {
            printf("词法分析第%d 行定义不符合规范!!! 出错词素：",lineCoun);
            cout<<word<<endl;
            outToken<<"词法分析第"<<lineCoun<<"行定义不符合规范!!! 出错词素："<<word<<endl;
            mark=0;
            while(line[i]!=' ')
                i++;
        }
      i++;
    }
    return mark;
}
scanner::scanner()
{

}

scanner::~scanner()
{
    //dtor
}
```

③语法分析函数

```
#include "parse.h"
#include "myclass.h"
#include <iostream>
#include <stack>
#include <iomanip>
#include <queue>
#include <fstream>
#define TerNUM 42
#define NTerNUM 44
using namespace std;

int wrongMark=0;

ofstream    outTrees("source/Tree.txt",ios::out);   ///产生的树写入 Tree.txt 中
ofstream    outStack("source/Stack.txt",ios::out);///将语法分析产生过程出入栈写入 Stack.txt
```

```
string terminal[TerNUM]={
                                    "BEGIN","END","if","else","while","int","float","ID","NUM","true","false","+","-","*","/","%","(",")",";",",",
                                    ">","<","==","filename","help","file","delay","reset","point","line","rotate","translate","scale",
                                    "get window size","set window size","set color","set point size","set line width","move to","line to","=",
                                  };

string nterminal[NTerNUM] ={
                                     "program","program2","Stmt-List","Stmt-List2","Stmt","Assign-Stmt","If-Stmt","If-Stmt2","Declare-Stmt",
                                     "ID-List","ID-List2","Repeat-Stmt","Compare-Exp","Command","Expr","Expr2","Term","Term2","Factor",

"ID-NUM","ID-NUM2","Add-OP","Multi-OP","Comp-OP","File-Com","File-Com2","Delay-Com","Set-Window-Com","Set-Color-Com",
                                     "Set-PointSiz-Com","Set-LineWid-Com","Point-Com","Scale-Com","Translate-Com","Reset-Com","Get-windsiz-Com",
                                     "MoveTo-Com","LineTo-Com","Line-Com","Rotate-Com", "Help-Com","Param-1","Param-2","Param-3"
                                    };

int findTerminal(token ttok){///查找终结符
    // cout<<"******************"<<"查找的词素为："<<ttok.index<<"种类："<<ttok.kind;
      int zz=-1;
      for(int i=0;i<TerNUM;i++)
       if(ttok.index==terminal[i])
          zz=i;
      //     cout<<"查找结果："<<zz<<endl;
      return zz;
}
int findNTerminal(token ttok){///查找非终结符
   for(int i=0;i<NTerNUM;i++){
      if(ttok.index==nterminal[i])
            return i;
   }
   return -1;
}

void parse::printStack(stack <token> ssta){    ///输出栈中的字符串
      cout<<"\n 当前栈为：";
      outStack<<"\n 当前栈为：";
      while(!ssta.empty())
      {
      cout<<ssta.top().index<<' ';
      outStack<<ssta.top().index<<' ';
      ssta.pop();
      }
      cout<<endl;
      outStack<<endl;
}
```

```
void gettoken(string gramm,token splitToken[],int &tokNum){
   int len = gramm.length();
   int i=0;
   while(i<len){
      int space=0;
      string ttoken="";
      while(i<len&&space<2){
         if(gramm[i]==' '){///空格忽略
              space++;
              if(ttoken=="get"||ttoken=="get window"||ttoken=="set"||ttoken=="set window"||ttoken=="set point"||ttoken=="set line"||ttoken=="move")
                   ttoken+=gramm[i];
              if(ttoken=="line"&&i+1<len&&gramm[i+1]=='t')
                   ttoken+=gramm[i];
         }
         else
            ttoken+=gramm[i],space=0;
         i++;
      }

      token temptok(ttoken, ttoken , ttoken ,-1);
      splitToken[tokNum++]=temptok;
   }
}



void buildTree(node *father,stack <node *> &tempsta){    ///建立树节点
        while(!tempsta.empty())
      {
         father->child[father->childNum] =    tempsta.top();
         father->childNum++;
         tempsta.pop();
      }
      return;
}

void pushstack(string gramm,stack <token> &sta,stack <node *> &nodeSta,node *father)///入栈建树过程
{
      int tokNum=0;
      token splitToken[10];
      gettoken(gramm,splitToken,tokNum);
      node *p;
      stack <node *> tempsta; ///用来暂存正向的树节点
      for(int i=tokNum-1; i>=0; i--)
      {
```

```
            p=new node();
            p->tokenn=splitToken[i];
            p->childNum=0;
            sta.push(splitToken[i]);
            nodeSta.push(p);
            tempsta.push(p);///放入暂存栈，用来之后建树
            cout<<"字符栈入栈"<<splitToken[i].index<<endl;
            outStack<<"字符栈入栈"<<splitToken[i].index<<endl;
        }
        buildTree(father,tempsta);
}

void parse::printTree (node *p) ///按照层输出建立的树
{
        queue <node *> que;
        que.push(p);
        outTrees<<"父节点  -->     孩子节点  \n\n";
        while(!que.empty())
        {
            node *top=que.front();
            que.pop();
            cout<<"节点"<<top->tokenn.index <<":" ;///        top->tokenn.index <<":" ;
            outTrees<<setiosflags(ios::fixed)<<setiosflags(ios::right)<<setw(12)<<top->tokenn.index <<"--> ";
            for(int i=0; i<top->childNum; i++)
            {
                if(top->child[i]->tokenn.index=="filename")
                    cout<<top->child[i]->tokenn.value<<" ";
                else
                    cout<<top->child[i]->tokenn.index<<' ';
                outTrees<<top->child[i]->tokenn.index<<"    ";
                if( findNTerminal( top->child[i]->tokenn )!=-1 )
                   que.push(top->child[i]);
            }

            if(top->childNum==0)
                cout<<"NULL";
            cout<<endl;
            outTrees<<endl;
        }
}


bool parse::function(token alltoken[],int tokenNum,stack <node * > &nodeSta,stack <token> sta,node * &root){
        int nowPoint=0;
        int terminalIndex=findTerminal(alltoken[nowPoint]); ///存储终结符的列号
```

```
token temptoken("program","program","",-1);
sta.push(temptoken);
cout<<"字符栈入栈"<<"pragram"<<endl;
outStack<<"字符栈入栈"<<"pragram"<<endl;

node *p;
p=new node();
p->tokenn=temptoken;
p->childNum=0;
root=p;
nodeSta.push(p);

while(!sta.empty()&&nowPoint<=tokenNum )
{
    printStack(sta);
    if(sta.top().index=="@"){ ///如果是空串，直接将栈顶的退出
       nodeSta.top()->tokenn.value="@";
       sta.pop();

       nodeSta.pop();
    }
    else if(findTerminal(sta.top())!=-1)///栈里是终结符，直接判断并匹配
    {
        if(sta.top().index==alltoken[nowPoint].index) ///正确匹配
        {
            cout<<"字符出栈"<<sta.top().index<<endl;
            outStack<<"字符出栈"<<sta.top().index<<endl;
            nodeSta.top()->tokenn.name=alltoken[nowPoint].name;
            //else
            nodeSta.top()->tokenn.value=alltoken[nowPoint].value;
            nodeSta.top()->tokenn.index=alltoken[nowPoint].index;
            nodeSta.top()->tokenn.loca=alltoken[nowPoint].loca;

            sta.pop();
            nodeSta.pop();


            nowPoint++;
            if(sta.empty())
                continue;
            terminalIndex=findTerminal(alltoken[nowPoint]); ///存储终结符的列号
        }
        else ///匹配出错
        {
```

```
                cout<<"\n\n\n【error】第"<<alltoken[nowPoint].loca<<"行的中间字符"<<alltoken[nowPoint].index<<"匹配错误"<<"期望的字符为："<<sta.top().index<<endl;
                return 0;
            }
        }
        else {   ///栈中为非终结符
            int notTerIndex = findNTerminal(sta.top());
            if(notTerIndex<0||notTerIndex>=45)   {
                cout<<"查找终结符的时候出错！！！查找的行数为："<<notTerIndex<<"该字符为："<<sta.top().index<<"种类："<<sta.top().name<<endl;
                return 0;
            }
            if( preTable[notTerIndex][ terminalIndex ] == "")     ///语法出现错误
            {
                // if
                cout<<alltoken[nowPoint].index<<"\n\n\n【error】语法错误，出现不被期望字符，错误位置第";
                if(alltoken[nowPoint].loca==0)   cout<<alltoken[nowPoint-1].loca ;
                else cout<<alltoken[nowPoint].loca ;
                cout<<"行,匹配错误的非终结符:";
                if(sta.top().index=="ID")   cout<<sta.top().name<<endl;
                else cout<<sta.top().value<<endl;
                break;
            }
            else if( preTable[notTerIndex][ terminalIndex ] == "@"){ ///出现空串，直接弹出后继续
                cout<<"字符出栈"<<sta.top().index<<endl;
                outStack<<"字符出栈"<<sta.top().index<<endl;

                node * father=nodeSta.top();
                sta.pop();
                nodeSta.pop();

                temptoken.name="@",temptoken.index="@",temptoken.value="",temptoken.loca=0;
                sta.push(temptoken); ///依然推到栈，便于之后统一操作

                p=new node();///建立新的树节点

                p->tokenn=temptoken ;
                p->childNum=0;
                nodeSta.push(p);
                father->child[father->childNum] = p;
                father->childNum++;
            }

            else{   ///预测分析表中不是空串
                cout<<"字符出栈"<<sta.top().index<<endl;
```

```
                    outStack<<"字符出栈"<<sta.top().index<<endl;
                    node *father=nodeSta.top();

                    nodeSta.pop();///将当前的非终结符推出
                    sta.pop();
                    pushstack(preTable[notTerIndex][ terminalIndex ],sta,nodeSta,father);   ///入栈的同时，建立树节点
                }
            }
        }
        if(nowPoint!=tokenNum){
           cout<<"匹配出错后出现剩余未匹配词素，且剩余词素为 ："<<endl;
           outTrees<<"\n=======================语法分析错误信息====================\n 剩余词素为：\n";
           for(int i=nowPoint;i<tokenNum;i++){
              printf("(%s,%s,%s,%d)\n",alltoken[i].index.c_str(),alltoken[i].name.c_str(),alltoken[i].value.c_str(),alltoken[i].loca);
              outTrees<<"("<<alltoken[i].index<<","<<alltoken[i].name<<","<<alltoken[i].value<<","<<alltoken[i].loca<<")"<<endl;
           }
           return false;
        }
        else if(!sta.empty()){
            cout<<"匹配完成时，栈中还有字符，语法错误\n 栈中剩余节点内容为："<<endl;
            outTrees<<"\n=======================语法分析错误信息====================\n 栈中剩余节点为：\n";
            while(!sta.empty()){
                printf("(%s,%s,%s,%d)\n",sta.top().index.c_str(),sta.top().name.c_str(),sta.top().value.c_str(),sta.top().loca);
                outTrees<<"("<<sta.top().index<<","<<sta.top().name<<","<<sta.top().value<<","<<sta.top().loca<<")";
                sta.pop();
            }
            return 0;
        }
        else {
                cout<<"语法分析结束，成功!!! "<<"\n 生成的语法树为：\n--------------------------------------------------------------------------------\n";
                outTrees<<"语法分析结束，成功!!! "<<"\n 生成的语法树为：\n--------------------------------------------------------------------------------\n";
                printTree(root);
                return 1;
        }
}


parse::parse()
{
    //ctor
}

parse::~parse()
{
    //dtor
```

```
}
```

④语义翻译函数

```
#include "trans.h"
#include "myclass.h"
#include <iostream>
#include <cstdio>
#include <stdlib.h>
#include <fstream>
using namespace std;

ofstream    outCommand("paint/paint.txt",ios::out);///将词法分析的分离出的词素写入 Token.txt 中
ofstream    outID("source/idTable.txt",ios::out);    ///输出变量表 idTable.txt

IDTable idTable;

int markDataType = 0;

void trans:: printResult(bool b){
if(b==1)    cout<<"\n\n=====================================编译成功=================================\n";
else    cout<<"\n\n=====================================编译失败=================================\n";
}

int findIDIndex(string str){
for(int i=0;i<idTable.idnum;i++)
      if(idTable.id[i].name==str)
         return i;
return -1;
}

string judgeIntOrFloat(string num){
      int len = num.length();
      for(int i=0;i<len;i++)
         if(num[i]=='.')
              return "float";
      return "int";
}

string trans:: getFloString(float flo)
{
      ofstream    outt("temp.txt",ios::out);
      ifstream inn("temp.txt");
      string temp;
      outt<<flo;
```

```
        outt.close();
        if(getline(inn,temp))
            return temp;
        else
        {
            cout<<"转换出现错误!!! 程序停止运行";
            printResult(0);
            exit(0);
        }
}

bool trans :: getStandarExp(node * root){    ///Term Expr2 || Factor Term2
    int i=0;//name
    int len = root->child[1]->tokenn.value.length() ;
    string str = root->child[1]->tokenn.value; ///后半部分运算式
    if(root->child[0]->tokenn.name=="float"||root->child[1]->tokenn.name=="float")
       for(int i=0;i<len;i++)
           if(str[i]=='%'){
               cout<<"【error】第"<<root->tokenn.loca<<"行语义翻译出现浮点数取余"<<endl;
               return 0;
           }
    float    x = atof(root->child[0]->tokenn.value.c_str());
    while(i<len){
        char op=str[i];
        i++;
       string temp="";
       int tem,strnu;
       while( i<len && (((str[i]<='9'&&str[i]>='0')||str[i]=='.')||(temp==""&&str[i]=='-')) )
           temp+=str[i++];
       switch(op){
        case '+': x=x+atof(temp.c_str()); break;
        case '-': x=x-atof(temp.c_str()); break;
        case '*': x=x*atof(temp.c_str()); break;
        case '/':
            if(temp=="0") {
                cout<<"【error】运行错误,分母为 0!!!  ";
                printResult(0);
                exit(0);
            }
            x=x/atof(temp.c_str()); break;
        case '%':
            tem=(int)x,strnu=atof(temp.c_str());
            x=tem%strnu; break;
       }
   }
```

```
    root->tokenn.name=((root->child[0]->tokenn.name=="float"||root->child[1]->tokenn.name=="float")?"float":"int");
    root->tokenn.value=getFloString(x);
    return 1;
}

void trans :: printID()
{
     for(int i=0; i<idTable.idnum; i++)
          outID<<idTable.id[i].name<<"         "<<idTable.id[i].value<<"         "<<idTable.id[i].intOrFlo<<endl;
     outID<<"========================================="<<endl<<endl;
}

void getCompExp(node *&root){      ///Expr Comp-OP Expr | true |false



     string leftExpTyp = root->child[0]->tokenn.name;    ///左面表达式值类型
     string righExpTyp = root->child[2]->tokenn.name;    ///左面表达式值类型
     if(leftExpTyp != righExpTyp)
       cout<<"【wearning】第"<<root->tokenn.loca<<"行 :条件语句中条件比较时"<<root->child[1]->tokenn.value<<"号左右比较值类型不同,
已强制类型转换!! \n" ;

  /// > 或者 <
          float leftValue = atof(root->child[0]->tokenn.value.c_str());
          float rightValue = atof(root->child[2]->tokenn.value.c_str());

          if(root->child[1]->tokenn.value == "<")
               (leftValue < rightValue) ? (root->tokenn.value = "true") : root->tokenn.value ="false";
          else if(root->child[1]->tokenn.value == ">")
               (leftValue > rightValue) ? root->tokenn.value= "true" : root->tokenn.value ="false";
          else{
               if((leftValue-rightValue)>-0.0000000001&&(leftValue-rightValue)<0.0000000001 )
                    root->tokenn.value = "true";
               else
                   root->tokenn.value = "false";
          }

}

void    trans::transFunction(node *&root){
if(root->tokenn.index=="program"){
     transFunction(root->child[1]);
}
else if(root->tokenn.index=="program2"){
     if(root->childNum>=2) ///Stmt-List end
     transFunction(root->child[0]);
```

```
}
else if(root->tokenn.index=="Stmt-List"){
    transFunction(root->child[0]);///Stmt
    transFunction(root ->child[1]);/// Stmt-List2
}
else if(root->tokenn.index=="Stmt-List2"){
    if(root->childNum>=2){ ///Stmt Stmt-List2
      transFunction(root->child[0]);
      transFunction(root->child[1]);
    }
}
else if(root->tokenn.index=="Stmt"){
    transFunction(root->child[0]); ///五种句型
    root->tokenn.loca=root->child[0]->tokenn.loca ;
}
else if(root->tokenn.index=="Assign-Stmt"){ ///    ID = Expr
    root->tokenn.loca=root->child[0]->tokenn.loca ;
    int idIndex= findIDIndex(root->child[0]->tokenn.name);
    if(idIndex==-1){
      printf("【error】第%d 行  变量%s 未经声明即使用，非法赋值!!! \n",root->tokenn.loca,root->child[0]->tokenn.name.c_str());
      printResult(0);
      exit(0);
    }
    transFunction(root->child[2]);
    if(idTable.id[idIndex].intOrFlo == root->child[2]->tokenn.name){ ///两种数据类型相同
          float xxx= atof(root->child[2]->tokenn.value.c_str());
          if(idTable.id[idIndex].intOrFlo=="int"&&xxx>0x7fffffff){
            cout<<"【wearning】第"<<root->tokenn.loca<<"行  :等式右边数值超过整型的最大值"<<endl;
          int temp = xxx;
          char ch[20] ;
          itoa(temp,ch,10);
          root->child[0]->tokenn.value = ch;
          }
          else{
          root->child[0]->tokenn.value = root->child[2]->tokenn.value;
          idTable.id[idIndex].value = root->child[2]->tokenn.value;
          }
    }
    else if( idTable.id[idIndex].intOrFlo == "float"    &&    root->child[2]->tokenn.name == "int" ){    ///float = int
            cout<<"【wearning】第"<<root->tokenn.loca<<"行  :赋值语句中=左右数据类型不相同，已强制类型转换"<<endl;
        root->child[0]->tokenn.value = root->child[2]->tokenn.value;
        idTable.id[idIndex].value = root->child[2]->tokenn.value;
    }
    else {     ///int = float ;
      cout<<"【wearning】第"<<root->tokenn.loca<<"行  :赋值语句中=左右数据类型不相同，已强制类型转换"<<endl;
```

```
            float xxx= atof(root->child[2]->tokenn.value.c_str());
            if(xxx>0x7fffffff)
                cout<<"【wearning】第"<<root->tokenn.loca<<"行 :等式右边数值超过整型的最大值"<<endl;
            int temp = xxx;
            char str[100];
            itoa(temp,str,10);
            root->child[0]->tokenn.value = str;
            idTable.id[idIndex].value    = str;
        }
    printID();
}
else if(root->tokenn.index=="If-Stmt"){    ///if ( Compare-Exp ) Stmt-List If-Stmt2
        transFunction(root->child[2]);
        root->tokenn.loca=root->child[0]->tokenn.loca ;
        if(root->child[2]->tokenn.value == "true")
            transFunction(root->child[4]);
        else {
            if(root->child[5]->childNum==3) ///If-stmt2
               transFunction(root->child[5]);
        }
}
else if(root->tokenn.index=="If-Stmt2"){    ///end | else Stmt-List end
        if(root->childNum==3){    ///else Stmt-List end
            transFunction(root->child[1]);
            root->tokenn.loca=root->child[1]->tokenn.loca ;
        }
}
else if(root->tokenn.index=="Declare-Stmt"){ /// int ID-List ; | float ID-List;
        root->tokenn.loca=root->child[0]->tokenn.loca ;
        if(root->child[0]->tokenn.value == "int"){
            markDataType=0;
            transFunction(root->child[1]);
        }
        if(root->child[0]->tokenn.value == "float"){
            markDataType=1;
            transFunction(root->child[1]);
        }
}
else if(root->tokenn.index=="ID-List"){ /// ID ID-List2
        if(findIDIndex(root->child[0]->tokenn.name)==-1){
        if(markDataType==0){    ///整数类型
            idTable.id[idTable.idnum].name=root->child[0]->tokenn.name;
            idTable.id[idTable.idnum].value = "";
            idTable.id[idTable.idnum++].intOrFlo="int";
```

```
            root->child[0]->tokenn.value = "";   ///此处无实际意义///整数类型值设置为未初始化
            transFunction(root->child[1]);
            root->child[1]->tokenn.value=="@"?(root->tokenn.loca=root->child[0]->tokenn.loca):(root->tokenn.loca=root->child[1]->tokenn.loca);
        }
        else{   ///浮点类型
            idTable.id[idTable.idnum].name=root->child[0]->tokenn.name;
            root->child[0]->tokenn.value = "";   ///浮点类型值设置为未初始化
            idTable.id[idTable.idnum++].intOrFlo="float";

            root->child[0]->tokenn.value = "";///此处无实际意义
            transFunction(root->child[1]);
            root->child[1]->tokenn.value=="@"?(root->tokenn.loca=root->child[0]->tokenn.loca):(root->tokenn.loca=root->child[1]->tokenn.loca);
            }
        }
        else {cout<<"【wearning】第"<<root->child[0]->tokenn.loca<<"行  声明变量语句中变量重复声明，重复声明无效!!! \n";}
}
else if(root->tokenn.index=="ID-List2"){ ///, ID ID-List2 | ε
        if(root->childNum!=1)   ///不为空串
          { ///, ID ID-List2
            if(markDataType==0){   ///整数类型
            idTable.id[idTable.idnum].name=root->child[1]->tokenn.name;
            idTable.id[idTable.idnum].value = "";
            idTable.id[idTable.idnum++].intOrFlo = "int";

            root->child[1]->tokenn.value = "";   ///整数类型值设置为未初始化   ///此处无实际意义
            root->tokenn.loca = root->child[1]->tokenn.loca;
            root->tokenn.name = "int";
            transFunction(root->child[2]);
            root->child[2]->tokenn.value=="@"?(root->tokenn.loca=root->child[1]->tokenn.loca):(root->tokenn.loca=root->child[2]->tokenn.loca);
            }
            else{   ///浮点类型
              idTable.id[idTable.idnum].name=root->child[1]->tokenn.name;
              idTable.id[idTable.idnum].value = "";
              idTable.id[idTable.idnum].intOrFlo = "float";

              root->child[1]->tokenn.value = "";   ///浮点类型值设置为未初始化   ///此处无实际意义
              root->tokenn.loca = root->child[1]->tokenn.loca;
              root->tokenn.name = "float";
              transFunction(root->child[2]);
              root->child[2]->tokenn.value=="@"?(root->tokenn.loca=root->child[1]->tokenn.loca):(root->tokenn.loca=root->child[2]->tokenn.loca);
            }
        }
        else{
            root->tokenn.value = "@";
        }
```

```
}
else if(root->tokenn.index=="Repeat-Stmt"){ ///while ( Compare-Exp ) Stmt-List end
	root->tokenn.loca=root->child[0]->tokenn.loca;
	transFunction(root->child[2]);
	while(root->child[2]->tokenn.value=="true"){
		transFunction(root->child[4]);
		transFunction(root->child[2]);
	}
}
else if(root->tokenn.index=="Compare-Exp"){	///Expr Comp-OP Expr | true |false
	if(root->childNum==1){
		if(root->child[0]->tokenn.value=="true")
			root->tokenn.value = "true", root->tokenn.name="KEYWORD"	,root->tokenn.loca=root->child[0]->tokenn.loca;
		else if(root->child[0]->tokenn.value=="false")
			root->tokenn.value = "false", root->tokenn.name="KEYWORD"	,root->tokenn.loca=root->child[0]->tokenn.loca;
	}
	else	///Expr Comp-OP Expr
	{
		transFunction(root->child[0]);
		transFunction(root->child[1]);
		transFunction(root->child[2]);
		root->tokenn.loca=root->child[2]->tokenn.loca;
		getCompExp(root);
	}
}
else if(root->tokenn.index=="Command"){
	transFunction(root->child[0]);
	root->tokenn.loca = root->child[0]->tokenn.loca;
}
else if(root->tokenn.index=="Expr"){	///Term Expr2
	transFunction(root->child[0]);
	transFunction(root->child[1]);
	root->tokenn.loca = root->child[0]->tokenn.loca;
	if(root->child[0]->tokenn.name=="float"||root->child[1]->tokenn.name=="float")
		root->tokenn.name = "float";
	  else root->tokenn.name="int";

	if(root->child[1]->tokenn.value=="@"){ ///第二个表达式为空
		root->tokenn.loca = root->child[0]->tokenn.loca;
		root->tokenn.value = root->child[0]->tokenn.value;
	}

	else {///Expr2 不为空
	  if(getStandarExp(root)==0) {
```

```
                printResult(0);
                  exit(0);
            }
        }
}
else if(root->tokenn.index=="Expr2"){    ///得到的表达式形式为 +8-9+10...
        if(root->childNum==3){    ///Add-OP Term Expr2
            transFunction(root->child[0]);
            transFunction(root->child[1]);
            transFunction(root->child[2]);
            root->tokenn.loca = root->child[1]->tokenn.loca;

            if(root->child[1]->tokenn.name=="float"||root->child[2]->tokenn.name=="float")
              root->tokenn.name = "float";
            else root->tokenn.name="int";

            root->tokenn.value = root->child[0]->tokenn.value + root->child[1]->tokenn.value ;
            if(root->child[2]->tokenn.value!="@"){    ///第三个表达式不为空
                root->tokenn.loca = root->child[2]->tokenn.loca;
                root->tokenn.value = root->tokenn.value+root->child[2]->tokenn.value;
                if(root->child[2]->tokenn.name == "float")    root->tokenn.name = "float";
            }
        }
        else
            root->tokenn.value="@";
}
else if(root->tokenn.index=="Term"){ //Factor Term2
        transFunction(root->child[0]);
        transFunction(root->child[1]);
        if(root->child[0]->tokenn.name=="float"||root->child[1]->tokenn.name=="float")
            root->tokenn.name = "float";
        else root->tokenn.name="int";
        root->tokenn.loca = root->child[0]->tokenn.loca;
        if(root->child[1]->tokenn.value=="@"){
            root->tokenn.value = root->child[0]->tokenn.value;
        }
        else ///term2 不为空
            if(getStandarExp(root)==0) {printResult(0);    exit(0);} ///获得当前节点值
}
else if(root->tokenn.index=="Term2"){    ///Multi-OP Factor Term2     得到的表达式形式为 *8/9*10...
        if(root->childNum==3){    ///Multi-OP Factor Term2
            transFunction(root->child[0]);
            transFunction(root->child[1]);
            transFunction(root->child[2]);
            root->tokenn.loca = root->child[0]->tokenn.loca;
```

```
            root->tokenn.value = root->child[0]->tokenn.value + root->child[1]->tokenn.value;
            root->tokenn.name = root->child[1]->tokenn.name;
            if(root->child[2]->tokenn.value!="@"){
                root->tokenn.loca = root->child[2]->tokenn.loca;
                root->tokenn.value = root->tokenn.value+root->child[2]->tokenn.value;
                if(root->child[2]->tokenn.name == "float")    root->tokenn.name = "float";
            }
        }
        else
            root->tokenn.value="@";
}
else if(root->tokenn.index=="Factor"){
        if(root->childNum==3){    ///( Expr )
            transFunction(root->child[1]);
            root->tokenn.loca = root->child[0]->tokenn.loca;
            root->tokenn.value = root->child[1]->tokenn.value;
            root->tokenn.name = root->child[1]->tokenn.name;
        }
        else{ ///    ID-NUM
            transFunction(root->child[0]);
            root->tokenn.loca = root->child[0]->tokenn.loca;
            root->tokenn.value = root->child[0]->tokenn.value;
            root->tokenn.name = root->child[0]->tokenn.name;
        }
}
else if(root->tokenn.index=="ID-NUM"){
    if(root->child[0]->tokenn.index=="ID"){      ///ID
        root->tokenn.loca = root->child[0]->tokenn.loca;
        int index = findIDIndex(root->child[0]->tokenn.name);
        if(index==-1){    ///未查找到该变量
            cout<<"【error】第"<<root->tokenn.loca<<"行  该引用变量未声明"<<endl;
            printResult(0);
            exit(0);
        }
        else {///查找到该变量
                if(idTable.id[index].value == ""){
                    cout<<"【wearning】第"<<root->tokenn.loca<<"行  变量"<<idTable.id[index].name<<"未经初始化,编译器已将该变量的值默认为 0，但其任未初始化"<<endl;
                    root->tokenn.value = "0";
                    ///idTable.id[index];
                }
            else ///能查找到该变量，且已经初始化
                root->tokenn.value = idTable.id[index].value;
        }
        root->tokenn.name = idTable.id[index].intOrFlo;
```

```
    }
    else if(root->child[0]->tokenn.value=="-"){    ///- ID-NUM2
        transFunction(root->child[1]);
        root->tokenn.loca = root->child[1]->tokenn.loca;
        root->tokenn.value = "-"+root->child[1]->tokenn.value;
        root->tokenn.name =    root->child[1]->tokenn.name;
     }
    else{    /// NUM
        root->tokenn.loca = root->child[0]->tokenn.loca;
        root->tokenn.value = root->child[0]->tokenn.value;

        root->tokenn.name = judgeIntOrFloat(root->child[0]->tokenn.value);
    }
}
else if(root->tokenn.index=="ID-NUM2"){    ///NUM | ID

     root->tokenn.loca = root->child[0]->tokenn.loca;
     if(root->child[0]->tokenn.index=="ID"){      ///ID
        int index = findIDIndex(root->child[0]->tokenn.name);
        root->tokenn.name = idTable.id[index].intOrFlo;

        if(index==-1){    ///未查找到该变量
            cout<<"【error】第"<<root->tokenn.loca<<"行  该引用变量未声明"<<endl;
            printResult(0);
            exit(0);
        }
        else {///查找到该变量
              if(idTable.id[index].value == ""){
                  cout<<"【wearning】第"<<root->tokenn.loca<<"行  变量"<<idTable.id[index].name<<"未经初始化,编译器已将该变量的值默认为 0，但其仍未初始化"<<endl;
                  root->tokenn.value = "0";
              }
             else ///能查找到该变量，且已经初始化
                 root->tokenn.value = idTable.id[index].value;                    ///部分强制转换未完成
        }
    }
    else{     ///NUM
        root->tokenn.name =    judgeIntOrFloat(root->child[0]->tokenn.value);
        root->tokenn.value = root->child[0]->tokenn.value;
     }
}
else if(root->tokenn.index=="Add-OP"){
     root->tokenn.value=root->child[0]->tokenn.value;
     root->tokenn.loca = root->child[0]->tokenn.loca;
}
```

```
else if(root->tokenn.index=="Multi-OP"){
    root->tokenn.value=root->child[0]->tokenn.value;
    root->tokenn.loca = root->child[0]->tokenn.loca;
}
else if(root->tokenn.index=="Comp-OP"){
    root->tokenn.value=root->child[0]->tokenn.value;
    root->tokenn.loca = root->child[0]->tokenn.loca;
}
else if(root->tokenn.index=="File-Com"){    /// file File-Com2
    transFunction(root->child[1]);
    root->tokenn.loca = root->child[0]->tokenn.loca;
    if(root->child[1]->tokenn.value !="" )    ///  有路径名称
        outCommand<<root->child[0]->tokenn.value<<":"<<root->child[1]->tokenn.value<<";"<<endl;    ///向文件张中输入 file 命令和文件路径
    else
        outCommand<<root->child[0]->tokenn.value<<";"<<endl;
}
else if(root->tokenn.index=="File-Com2"){
    root->tokenn.loca = root->child[0]->tokenn.loca;
    if(root->child[0]->tokenn.index == "filename"){
        root->tokenn.value=root->child[0]->tokenn.value;
    }
    else
        root->tokenn.value ="";
}
else if(root->tokenn.index=="Delay-Com"){    ///delay Param-1
    transFunction(root->child[1] );
    root->tokenn.loca = root->child[0]->tokenn.loca;
    if(root->child[1]->tokenn.value=="")
        outCommand<<"delay"<<";"<<endl;
    else
        outCommand<<"delay "<< root->child[1]->tokenn.value <<";"<<endl;
}
else if(root->tokenn.index=="Set-Window-Com"){    ///set window size Param-1
    transFunction(root->child[1] );
    root->tokenn.loca = root->child[0]->tokenn.loca;
    if(root->child[1]->tokenn.value=="")
        outCommand<<"set window size"<<";"<<endl;
    else
        outCommand<<"set window size "<< root->child[1]->tokenn.value <<";"<<endl;
}
else if(root->tokenn.index=="Set-Color-Com"){    ///set color Param-3
    transFunction(root->child[1] );
    root->tokenn.loca = root->child[0]->tokenn.loca;
    outCommand<<"set color "<<root->child[1]->tokenn.value<<";"<<endl;
}
```

```
else if(root->tokenn.index=="Set-PointSiz-Com"){    ///    set point size Param-1
        transFunction(root->child[1] );
        root->tokenn.loca = root->child[0]->tokenn.loca;
        if(root->child[1]->tokenn.value=="")
                outCommand<<"set point size"<<";"<<endl;
        else
                outCommand<<"set point size "<< root->child[1]->tokenn.value <<";"<<endl;
}
else if(root->tokenn.index=="Set-LineWid-Com"){    ///      set line width Param-1
        transFunction(root->child[1] );
        root->tokenn.loca = root->child[0]->tokenn.loca;
        if(root->child[1]->tokenn.value=="")
                outCommand<<"set point size"<<";"<<endl;
        else
                outCommand<<"set point size "<< root->child[1]->tokenn.value <<";"<<endl;
}
else if(root->tokenn.index=="Point-Com"){    ///point Param-2
        transFunction(root->child[1] );
        root->tokenn.loca = root->child[0]->tokenn.loca;
        if(root->child[1]->tokenn.value=="")
                outCommand<<"point"<<";"<<endl;
        else
                outCommand<<"point "<< root->child[1]->tokenn.value <<";"<<endl;
}
else if(root->tokenn.index=="Scale-Com"){    ///scale Param-2
        transFunction(root->child[1] );
        root->tokenn.loca = root->child[0]->tokenn.loca;
        if(root->child[1]->tokenn.value=="")
                outCommand<<"scale"<<";"<<endl;
        else
                outCommand<<"scale "<< root->child[1]->tokenn.value <<";"<<endl;
}
else if(root->tokenn.index=="Translate-Com"){    ///translate Param-2
        transFunction(root->child[1] );
        root->tokenn.loca = root->child[0]->tokenn.loca;
        if(root->child[1]->tokenn.value=="")
                outCommand<<"translate"<<";"<<endl;
        else
                outCommand<<"translate "<< root->child[1]->tokenn.value <<";"<<endl;
}
else if(root->tokenn.index=="Reset-Com"){    ///reset
        root->tokenn.loca = root->child[0]->tokenn.loca;
        outCommand<<"reset"<<";"<<endl;
}
else if(root->tokenn.index=="Get-windsiz-Com"){    ///get window size
```

```
        root->tokenn.loca = root->child[0]->tokenn.loca;
        outCommand<<"get window size"<<";"<<endl;
}
else if(root->tokenn.index=="MoveTo-Com"){    ///move to ID-NUM,ID-NUM
        transFunction(root->child[1] );
        transFunction(root->child[3] );
        root->tokenn.loca = root->child[3]->tokenn.loca;
        outCommand<<"move to "<<root->child[1]->tokenn.value<<","<<root->child[3]->tokenn.value<<";"<<endl;
}
else if(root->tokenn.index=="LineTo-Com"){    ///line to ID-NUM,ID-NUM
        transFunction(root->child[1] );
        transFunction(root->child[3] );
        root->tokenn.loca = root->child[3]->tokenn.loca;
        outCommand<<"line to "<<root->child[1]->tokenn.value<<","<<root->child[3]->tokenn.value<<";"<<endl;
}
else if(root->tokenn.index=="Line-Com"){    ///line ID-NUM,ID-NUM,ID-NUM,ID-NUM
        transFunction(root->child[1] );
        transFunction(root->child[3] );
        transFunction(root->child[5] );
        transFunction(root->child[7] );
        root->tokenn.loca = root->child[7]->tokenn.loca;
        outCommand<<"line
"<<root->child[1]->tokenn.value<<","<<root->child[3]->tokenn.value<<","<<root->child[5]->tokenn.value<<","<<root->child[7]->tokenn.value<<";
"<<endl;
}
else if(root->tokenn.index=="Rotate-Com"){    ///rotate ID-NUM
        transFunction(root->child[1] );
        root->tokenn.loca = root->child[1]->tokenn.loca;
        outCommand<<"rotate "<<root->child[1]->tokenn.value<<";"<<endl;
}
else if(root->tokenn.index=="Help-Com"){    ///help
        outCommand<<"help"<<";"<<endl;
        root->tokenn.loca = root->child[0]->tokenn.loca;
}
else if(root->tokenn.index=="Param-1"){
     if( root->child[0]->tokenn.index == "@" )
            root->tokenn.value="",root->tokenn.loca=-1;
     else
     {
        transFunction(root->child[0]);
        root->tokenn.value =    root->child[0]->tokenn.value;
        root->tokenn.loca = root->child[0]->tokenn.loca;
        if(root->child[0]->tokenn.name=="float")    root->tokenn.name="float";
        else    root->tokenn.name="int";
     }
```

```
}
else if(root->tokenn.index=="Param-2"){
	if( root->child[0]->tokenn.index =="@" )
		root->tokenn.value="",root->tokenn.loca=-1;
	else{
		transFunction(root->child[0]);
		transFunction(root->child[2]);
		root->tokenn.value = root->child[0]->tokenn.value+","+root->child[2]->tokenn.value;
		root->tokenn.loca = root->child[2]->tokenn.loca;
		if(root->child[0]->tokenn.name=="float"||root->child[2]->tokenn.name=="float")
			root->tokenn.name="float";
		else
			root->tokenn.name="int";
	}
}
else if(root->tokenn.index=="Param-3"){
    if( root->child[0]->tokenn.index =="@" )
		root->tokenn.value="",root->tokenn.loca=-1;
	else{
		transFunction(root->child[0]);
		transFunction(root->child[2]);
		transFunction(root->child[4]);
		root->tokenn.value = root->child[0]->tokenn.value+","+root->child[2]->tokenn.value+","+root->child[4]->tokenn.value;
		if(root->child[0]->tokenn.name=="float"||root->child[2]->tokenn.name=="float"||root->child[4]->tokenn.name=="float")
			root->tokenn.name="float";
		else
			root->tokenn.name="int";
		root->tokenn.loca = root->child[4]->tokenn.loca;
	}
}
else{
	cout<<"出错了,且想要查找的非终结符为："<<root->tokenn.index<<"第"<<root->tokenn.loca<<"行"<<endl;
	printResult(0);
	exit(0);
}
}

trans::trans()
{
   outID<<"==================变量表====================\nName value type\n\n";
}
trans::~trans()
{

}
```