语法分析器报告

摘要：

本次语义分析是在之前词法分析和 LR(1)语法分析的基础上修改的，主要修改部分在于加了一个 value_transmission 函数，使得在规约过程中可以将真实的数字或者变量名或者中间变量名进行传递，然后还加入了一个 Translate 函数,通过对当前分析出来的符号值和之前的符号值,来推测出现在分析出的语义，从而生成四元表达式。

本次作业在一般的要求上加入了 for 语句的语义分析和 return 语句的语义分析,以及对静态语义错误中的未定义变量的使用的检测,程序可处理能力较强，报错也较为明显，展示信息也使得较为充分了。

一、运行和开发环境

运行环境：window 10
开发环境：QT5.14.2

二、能识别的单词、能分析的语法、扩充的功能

1.能识别的保留字单词和运算符如下:

```
//全局变量，保留字表
static char reserveWord[WORDLEN][20] = {
    "include","using","namespace","std","define",
    "main","bool","auto", "break", "case", "char", "const", "continue",
    "cout","cin","_getch","default", "do", "double", "else", "enum", "extern", "while",
    "float", "for", "goto", "if", "int", "long","string",
    "register", "return", "short", "signed", "sizeof", "static",
    "struct", "switch", "typedef", "union", "unsigned", "void",
    "volatile","system"
};
//运算符表
static char oneOpera[ONEOPLEN] = {
'+', '-', '*', '/', '<',  '>',  '=',
';', '(', ')', '^', ',','\"', '\'', '#', '&',
'|',  '%', '~',  '[', ']', '{','\\',
'}' , '.', ':', '!','\?',
};
static char twoOpera[TWOOPLEN][3] = {
     "<=", ">=",  "==",
    "!=","&&","||","<<", ">>"
};
```

2. 能分析的语法

题目要求的全部：

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>‘)’

额外扩充的语法：
1. Program 可以包含多个函数
2. 添加了 for 语句. eg:

```
for(int i=0;i<8;i+=2)
{
    j=i+1;
    if(a>(b+c))
    {
        j=a+(b*c+1);
    }
    else
   {
       j=a;
   }
}
```

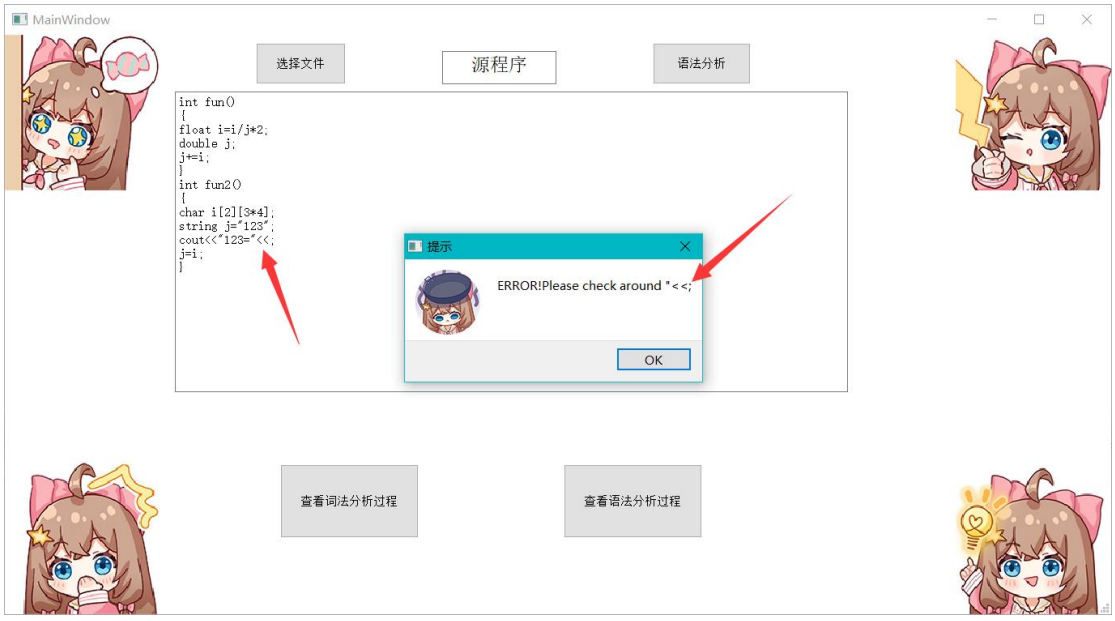
3. 定义语句可以直接赋值。eg：int a=0;
4. 添加了简便赋值语句。eg：a+=1;
5. 可以定义数组。eg：char i[2][3*4];
6. 可以使用字符串赋值 string。eg：string s="12345 Hello World"
7. 支持分析 cout 语句。eg：cout<<"The result of i+j is"<<i+j;

3.能够进行语义分析的语句

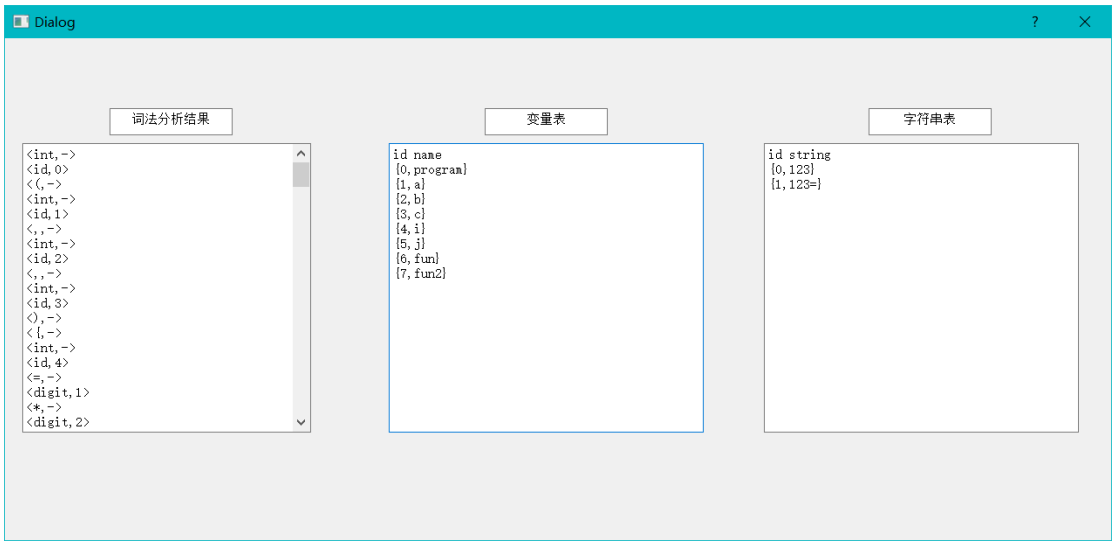
1.-i+k*(i*2.33-200),这种正常的表达式
2.i=j*2，这种赋值语句
2.i+=j，这种简便赋值
3.if 和 if{}else{}语句
4.while(){}语句
5.for(int i=0;i<10;i+=1)这种 for 语句
6.return i 这种 return 语句

3. 扩充的功能

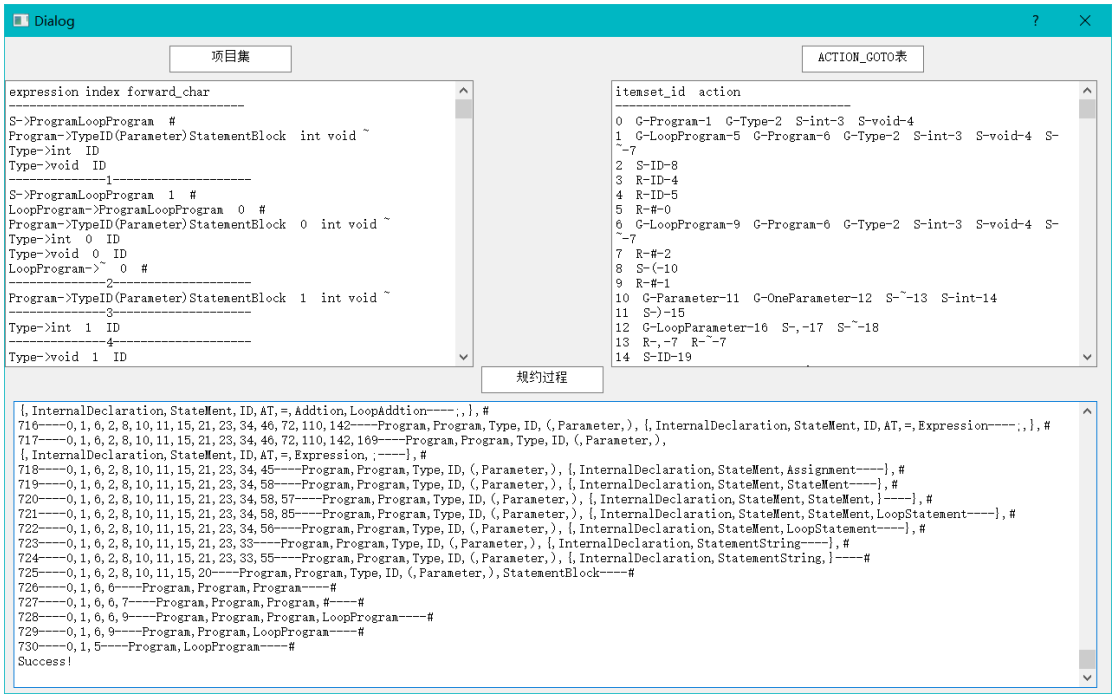
1.扩充了一部分如上说明的语法
2.具有一定的报错功能，能够将错误附件的单词显示出来。



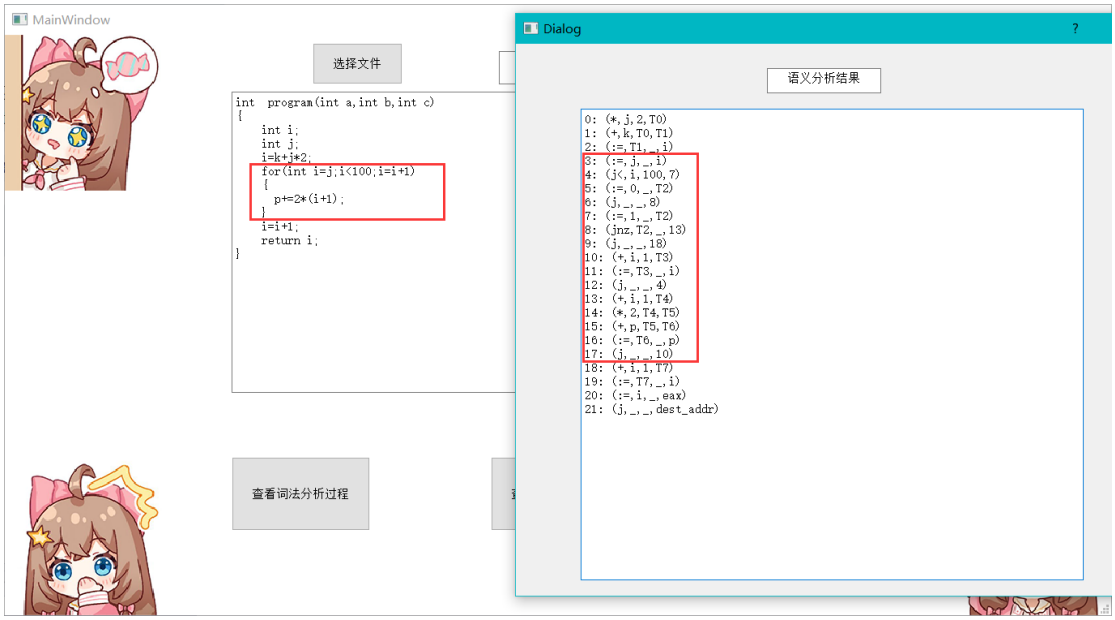
3.能够查看语法分析过程中产生的语法分析结果、变量表和字符串表



4. 能够查看语法分析过程中产生的所有项目集，和 ACTION_GOTO 表，以及整个规约过程。

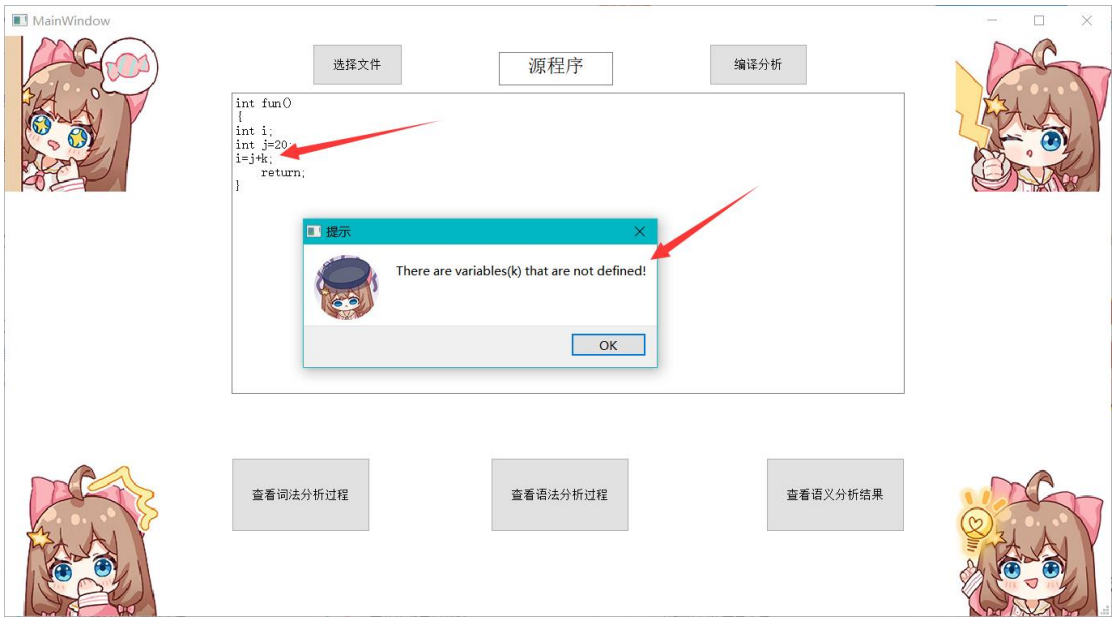


5. 扩充了 for 语句的语义分析



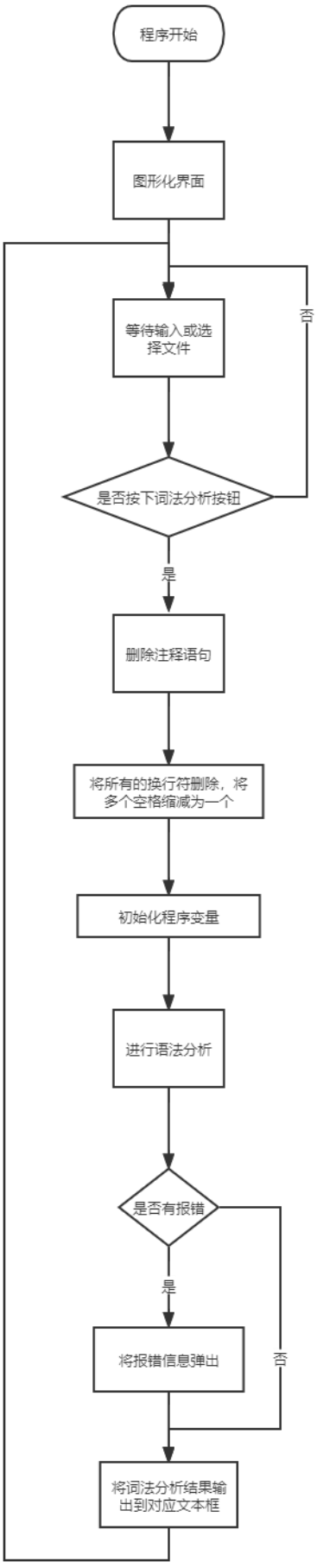
6. 扩充了 return 语句的分析

7. 能够识别使用了未定义的变量。

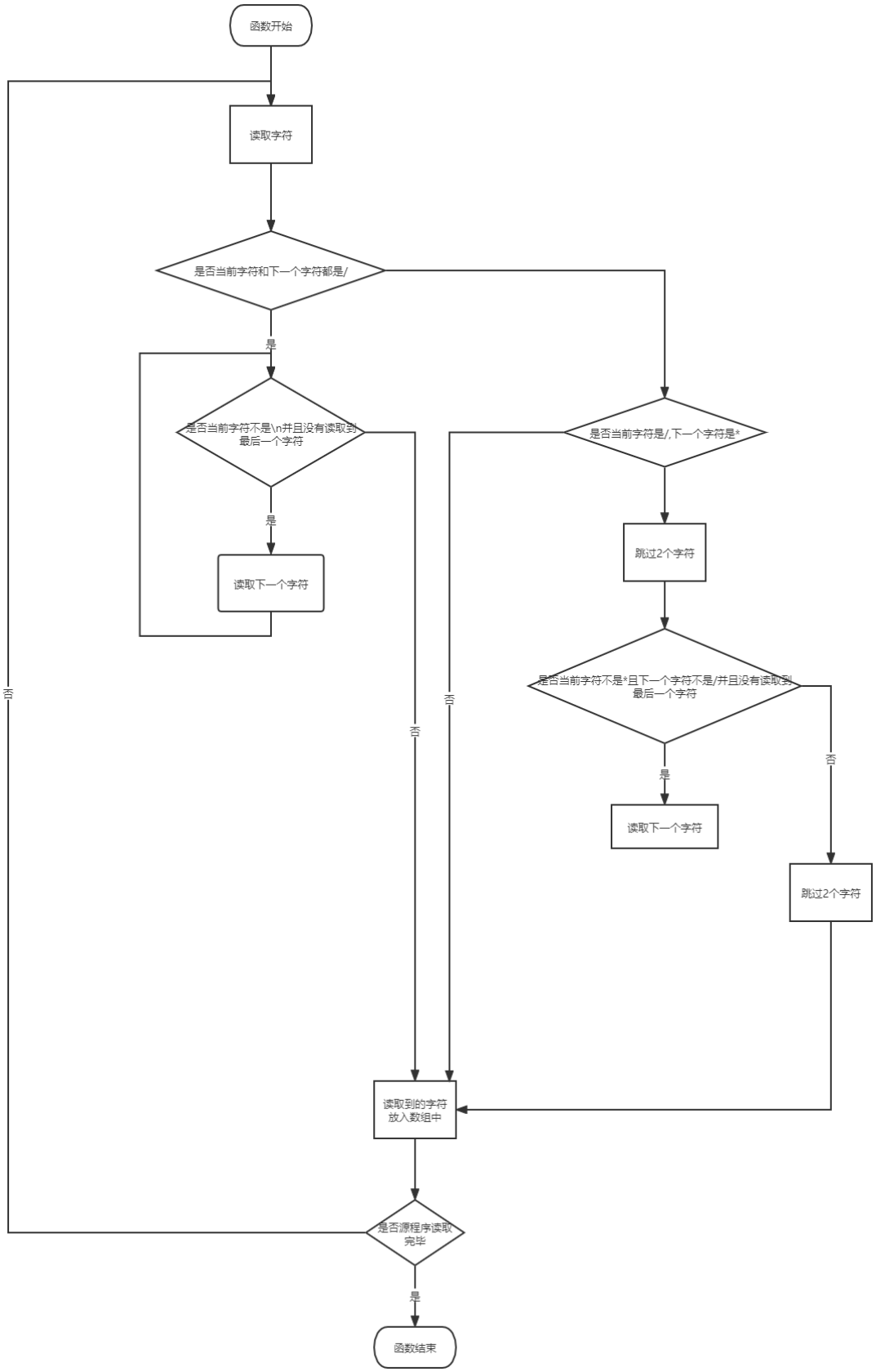


分析算法的主程序框图

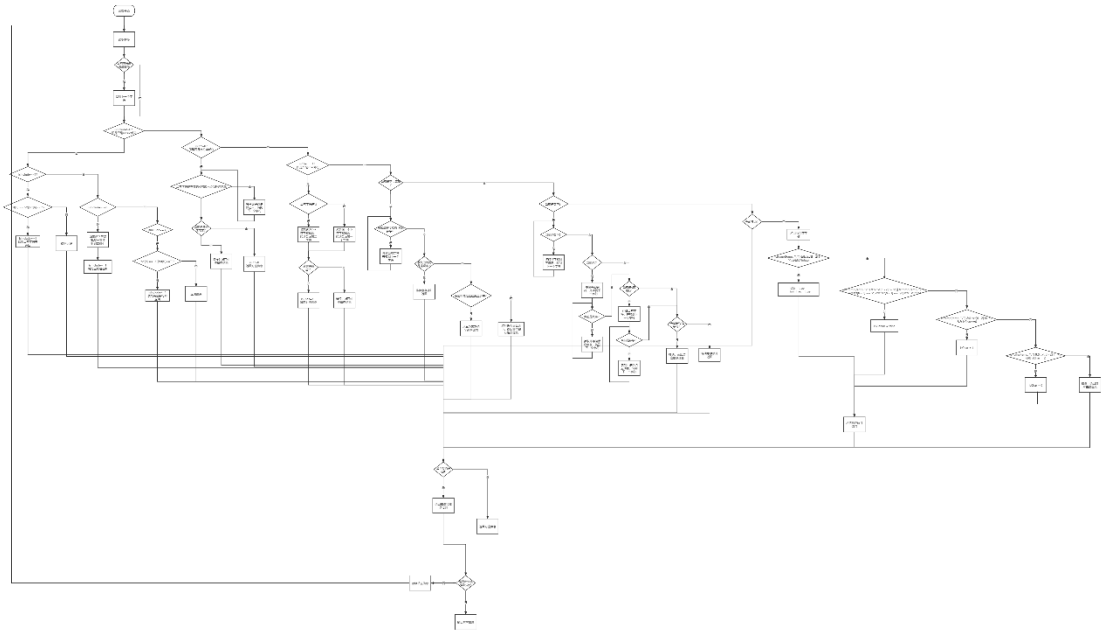
词法分析器部分



删除语句的函数流程图


函数开始
读取字符
是否当前字符和下一个字符都是/
是
是否当前字符不是\n并且没有读取到最后一个字符
是
读取下一个字符
否
是否当前字符是/,下一个字符是*
否
跳过2个字符
是否当前字符不是*且下一个字符不是/并且没有读取到最后一个字符
是
读取下一个字符
否
跳过2个字符
读取到的字符放入数组中
是否源程序读取完毕
否
是
函数结束

语法分析的函数流程图



（附件有清晰图片）

语法分析器部分

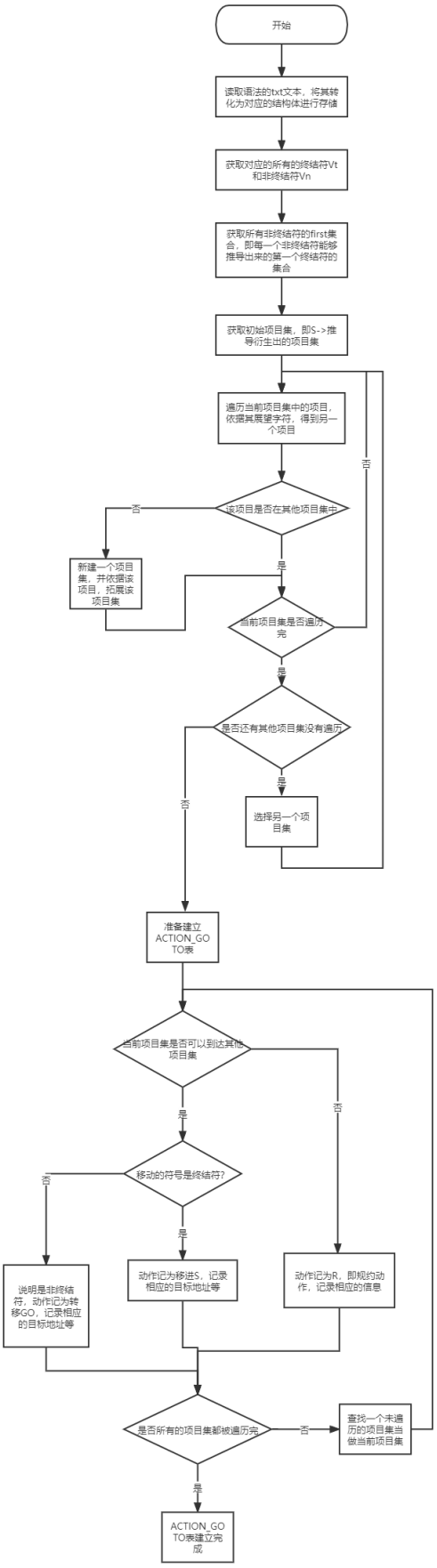
本次语义分析器和语法分析器的构建采用 class 类进行构建，如下：

```
class syntactic_parser
{
private :
    string* txt;//语法规则对应的文本
    int txt_len;//语法规则对应的文本的长度
    STR* p;//所有的语法规则
    int p_len;//语法规则的长度
    V* Vt;//所有的终结符
    V* Vn;//所有的非终结符
    int Vt_len;//终结符的数量
    int Vn_len;//非终结符的数量
    FF* first;//所有非终结符对应的first集合
    int first_len;//first集合的长度
    Itemset* itemsets;//所有的项目集
    int itemset_len=0;//所有项目集的长度
    ALL_ACTION_GOTO* aag;//ACTION_GOTO表
    V* aTxt;//待分析的程序经过词法分析器分析后得到的结果
    int* state;//状态表
    //id均指向当前元素
    int state_id;//状态表现在所指的状态id
    //int step;//
    V* Symbol;//符号表
    int Symbol_id;//符号表现在所指的符号id
    int txt_id;//读取了文本的符号id
    FourTranslate* ft;//四元式队列
    int ft_len;//四元式的长度
    int tran_temp_id;//临时变量T的id
    stack<int> fillback;//记录需要回填的表达式的id
    stack<int> rememberback;//记录后面需要用的位置
    string* define_var;//记录定义的变量名
    int define_var_len;//定义的变量的长度

    bool is_In_Vt(V a);//是否是终结符
    bool is_In_Vn(V a);//是否是非终结符
    void Letter_First(V a);//计算a的first集合
    bool isInFirst(FF a, V b);//b是否在a的first集合中
    bool isInV(V* a, int a_len, V b);//b是否在集合a中
    int getVnIndex(V a);//a在非终结符表中的位置
    int getSTRIndex(STR a);//得到a在语法规则表中的位置
    void getOtherItem(Itemset& a, int a_i);//得到项目集中的其他项目
    void getOtherForward(Itemset& a, int a_i, int n_i);//得到其他的展望集合
    void changeOnce(int i);//第i个项目集拓展出其他的项目集
    int isInOtherItemset(Item a);//是否在其他已有的项目集中
    bool isItemSame(Item a, Item b);//2个项目是否相同
    void copyItem(Item orign, Item& target);//复制一份项目
    int getAction(int state,V a,int& action,int& target);//根据ACTION_GOTO表得到在当前状态，当前字符下应产生的动作
    string errorMsgChange(V a);//报错信息的V的转化
    void value_transmission(int target);//传递变量名或数值
    void get_ft_res();//四元式转化为string
    bool IDIsDefine(string dvar);//变量是否被定义
    void Translate(int& synIsError,string& synErrorMsg);//转化为四元式
    void showFT();//展示四元式
public:
    int aTxt_len;//词法分析器分析后得到的结果的长度
    int is_lex_error;//语法分析是否有错
    string lex_res, lex_variable, lex_string_table, lex_error_msg;//词法分析产生词法分析的结果、变量表、 字符串表、词法分
    string syn_itemset, syn_action_goto, syn_process;//语法分析的产生的项目集、ACTION_GOTO表、语法分析的过程
    string ft_res;//四元式的字符串结果
    syntactic_parser();//建立的初始化
    void read_file(string path);//读取文件
    void getaTxt(char* txt);//调用词法分析器从文本中得到要分析的程序
    void get_Vt_Vn();//得到终结符和非终结符
    void get_First();//得到First集合
    void get_Itemset();//得到项目集
    void getInitItemset();//得到初始项目集
    void changeTogetOtherItem();//得到其他的项目集
    void createACTIONGO();//创建ACTION_GOTO表
    void Analysis(int& synIsError,string& synErrorMsg);//语法分析
    void init();//初始化一遍关键变量
};
```

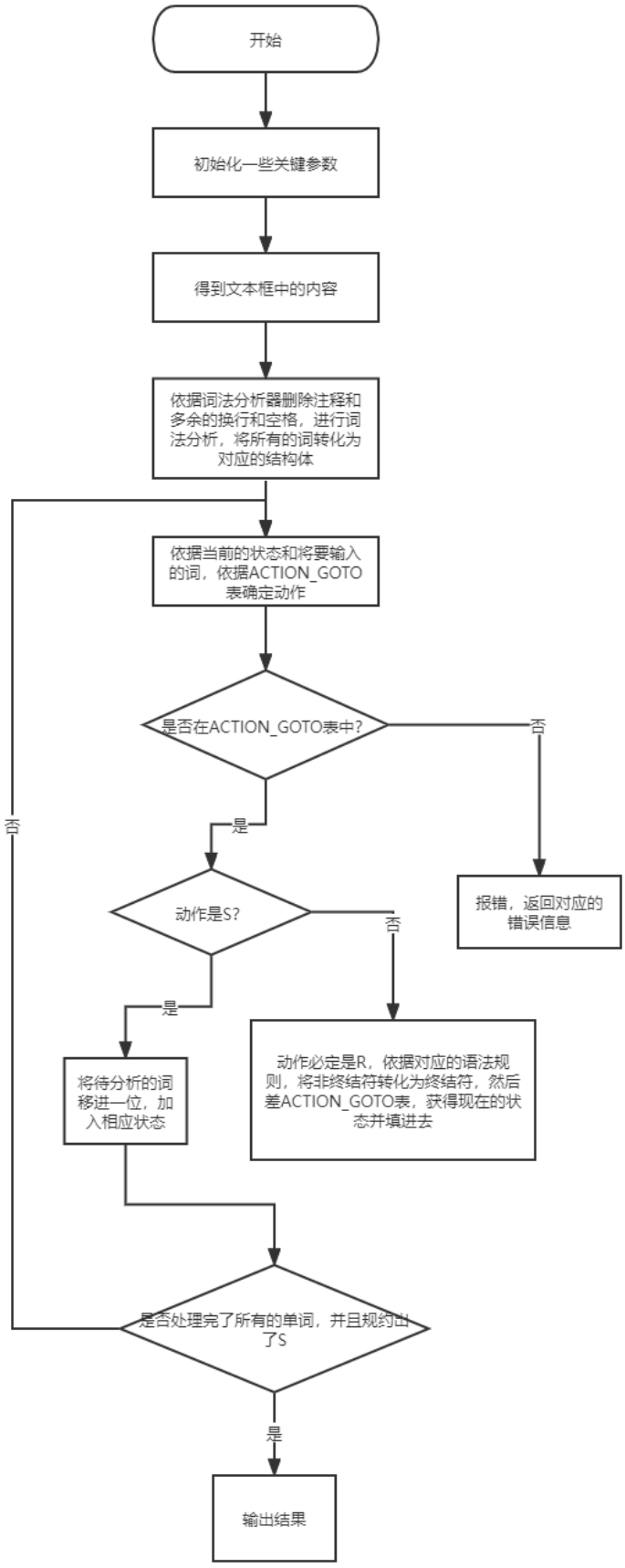
自定义了一些结构体：

```
struct V//终结符或非终结符
{
    string name;//符号名字
    bool type;//0代表非终结符，1 代表终结符
    string realName;//记录真实的变量名和字符串名
    bool operator==(const V b) const//重构==，判断2个V是否相等
    {
        return this->name == b.name && this->type == b.type;
    }
};
struct STR//推导式
{
    V left;//左边符号
    V right[MAX];//右边符号
    int right_len = 0;//右边符号的长度
};
struct FF//first集合
{
    V name;//对应的非终结符名字
    V first_list[MAX];//first列表
    int first_list_len = 0;//first列表的长度
};
struct Item//项目
{
    STR expression;//推导式
    V forward[LM];//展望字符
    int forward_len = 0;//展望字符长度
    int index = 0;//当前右部坐标
};

struct itemChange//项目的移动信息
{
    int target;//移动目标
    V method;//借助的符号
};

struct Itemset//项目集
{
    Item item[LM];//项目列表
    int item_len = 0;//项目列表长度
    itemChange change[LM];//该项目集转换到其他项目集的方式和目标
    int change_len = 0;//change数组的有效长度
};

struct ACTION_GOTO//ACTION_GOTO表的一个单元
{
    V method;//借助的符号
    int action;//动作，go:0 s:1 r:2
    int target;//目标
};

struct ALL_ACTION_GOTO//ACTION_GOTO表的一行
{
    ACTION_GOTO* ag;//这一行中的所有有效单元
    int len = 0;//这一行中的所有有效单元的长度
};
```

语法分析前需要依据语法规则建立好 ACTION_GOTO 表，流程如下：

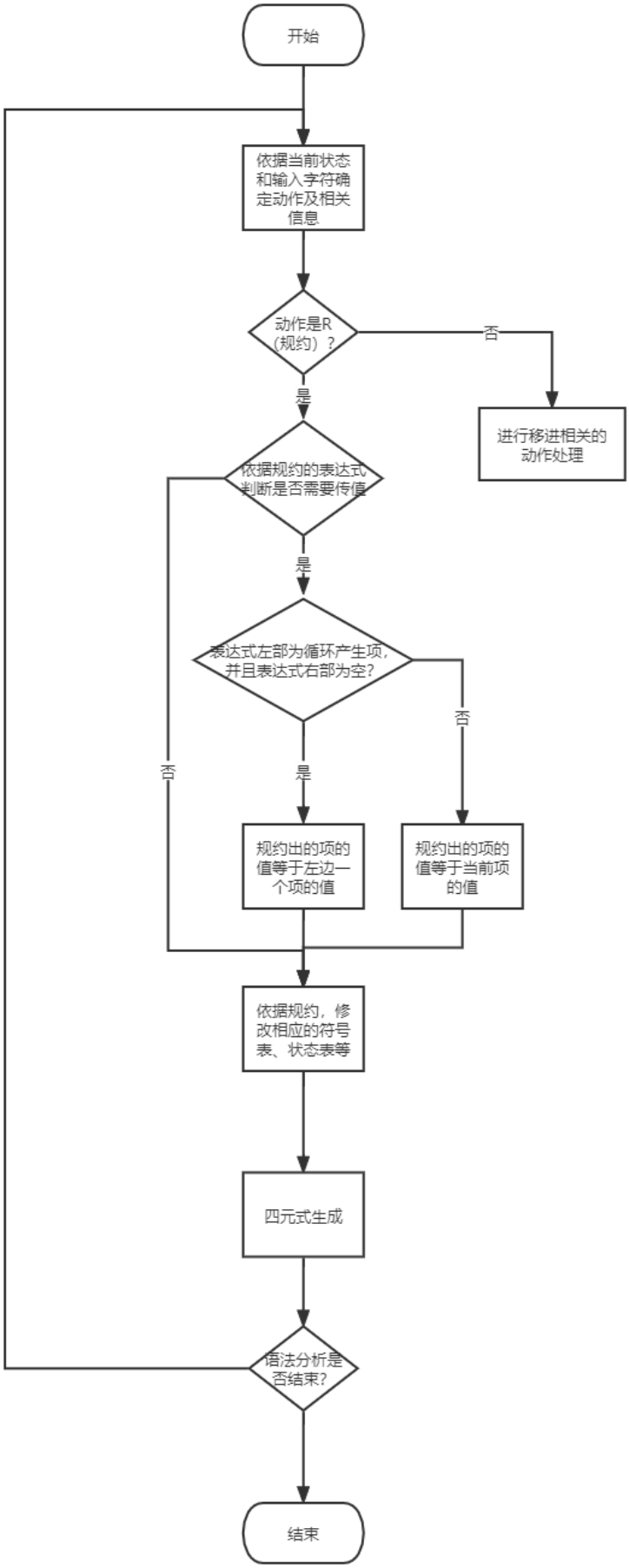


语法分析时的流程图如下：

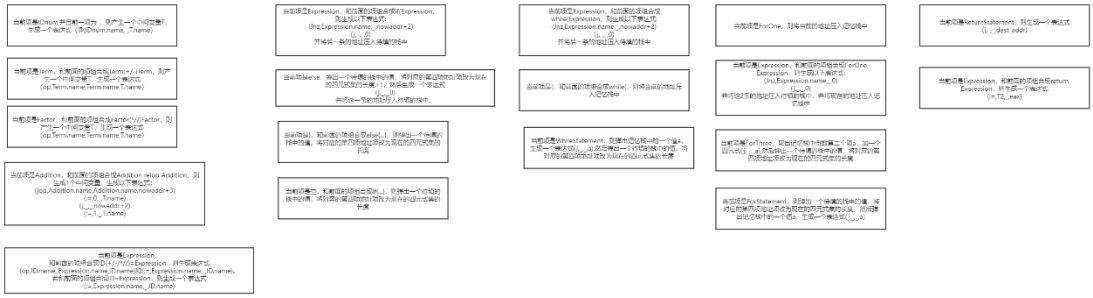


语义分析器部分

在规约过程中进行值的传递



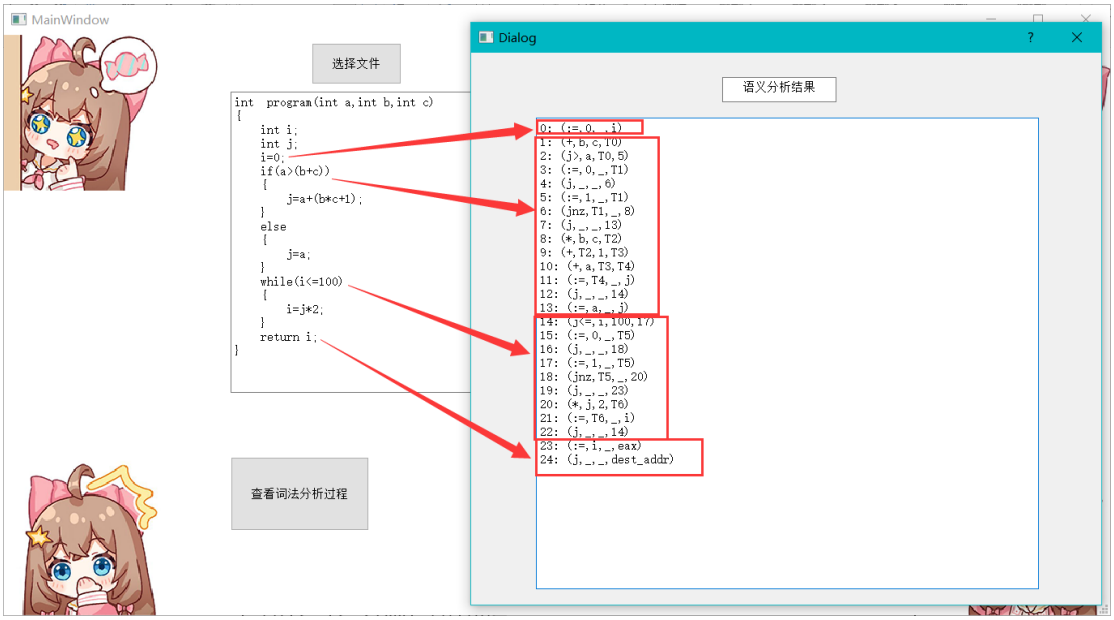
2.依据符号栈中的内容进行语义分析



（附件有清晰图片）

运行结果截图

PPT 示例程序：

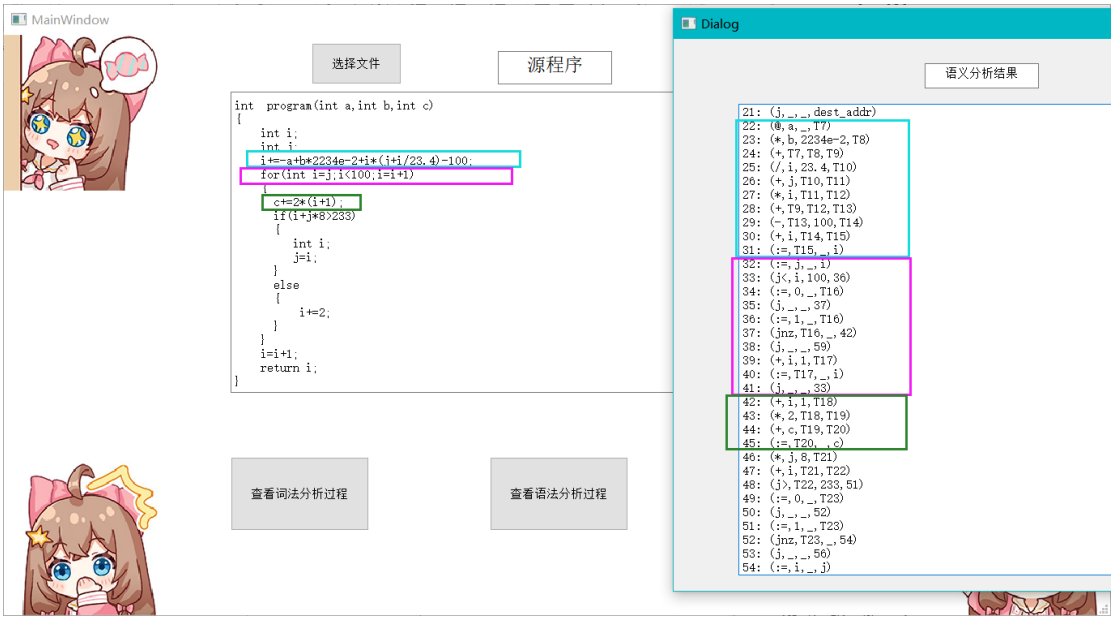


自己构建的较为复杂的程序分析结果:

程序代码：

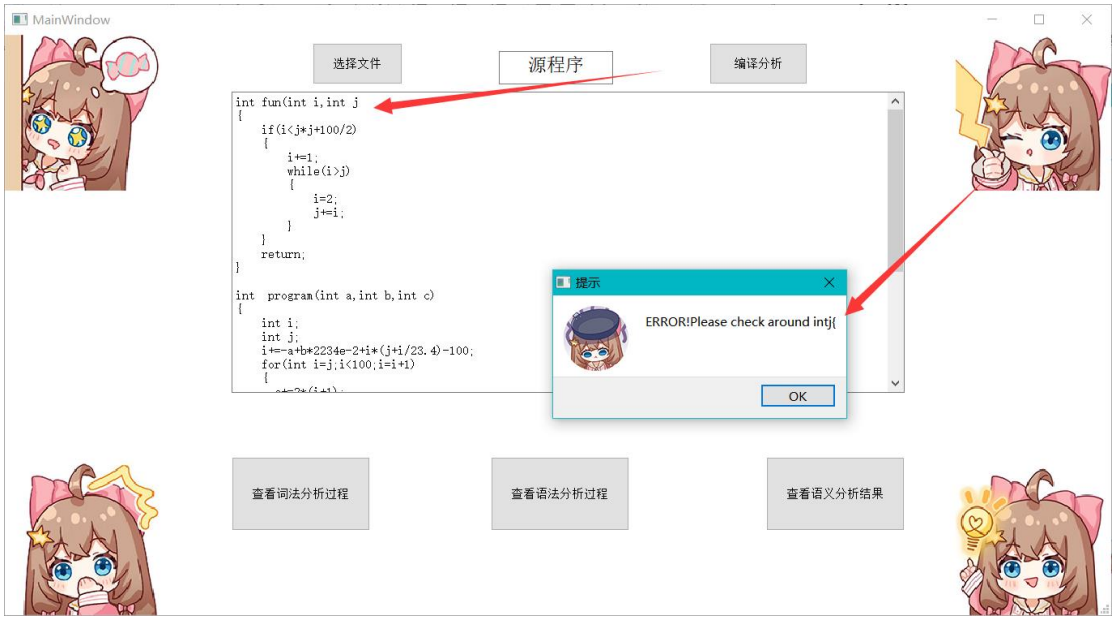
```
//test
int fun(int i,int j)
{
/*
There is something need to delete;
233333
*/
    if(i<j*j+100/2)
    {
        i+=1;
        while(i>j)
        {
            i=2;
            j+=i;
        }
    }
    return;
}

int  program(int a,int b,int c)
{
    int i;
    int j;
    i+=-a+b*2234e-2+i*(j+i/23.4)-100;
    for(int i=j;i<100;i=i+1)
    {
      c+=2*(i+1);
      if(i+j*8>233)
      {
         int i;
         j=i;
      }
      else
      {
          i+=2;
      }
    }
    i=i+1;
    return i;
}
```


MainWindow
选择文件
源程序
//test
int fun(int i,int j)
{
/*
There is something need to delete;
233333
*/
if(i<j*j+100/2)
{
i+=1;
while(i>j)
{
i+=2;
j+=i;
}
}
return;
}
int program(int a,int b,int c)
{
查看词法分析过程
查看语法分析过程
Dialog
语义分析结果
0: (*, j, j, T0)
1: (/, 100, 2, T1)
2: (+, T0, T1, T2)
3: (j<, i, T2, 6)
4: (:=, 0, _, T3)
5: (j, _, _, 7)
6: (:=, 1, _, T3)
7: (jnz, T3, _, 9)
8: (j, _, _, 21)
9: (+, i, 1, T4)
10: (:=, T4, _, i)
11: (j>, i, j, 14)
12: (:=, 0, _, T5)
13: (j, _, _, 15)
14: (:=, 1, _, T5)
15: (jnz, T5, _, 17)
16: (j, _, _, 21)
17: (:=, 2, _, i)
18: (+, j, i, T6)
19: (:=, T6, _, j)
20: (j, _, _, 11)
21: (j, _, _, dest_addr)
22: (@, a, _, T7)
23: (*, b, 2234e-2, T8)
24: (+, T7, T8, T9)
25: (/, i, 23.4, T10)
26: (+, j, T10, T11)
27: (*, i, T11, T12)
28: (+, T9, T12, T13)
29: (-, T13, 100, T14)
30: (+, i, T14, T15)
31: (:=, T15, _, i)
32: (:=, j, _, i)
33: (j<, i, 100, 36)

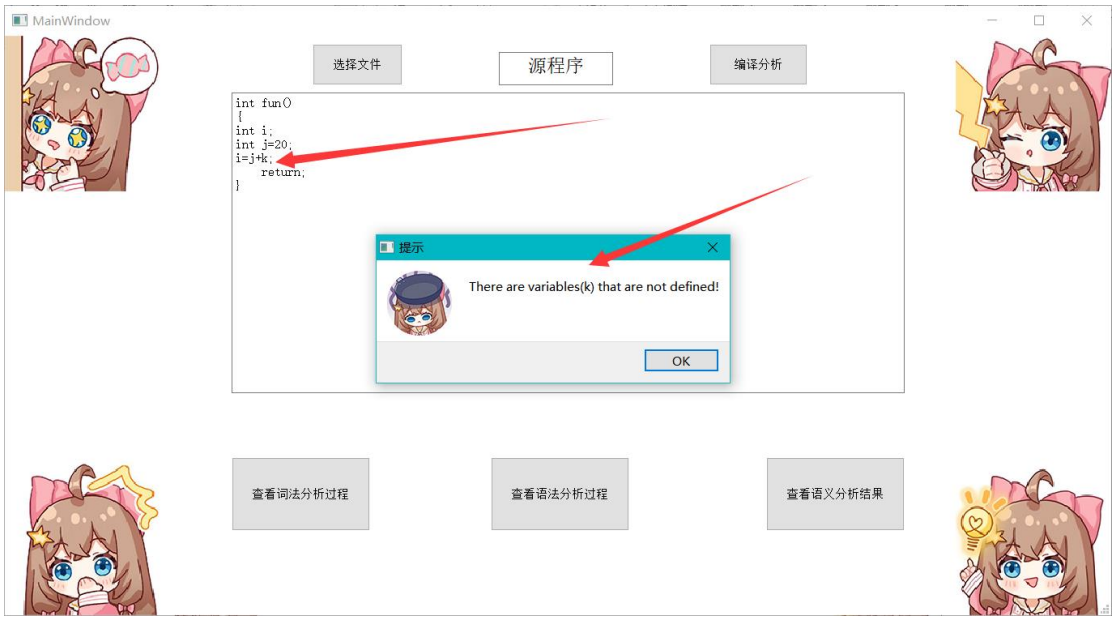

MainWindow
选择文件
源程序
int program(int a,int b,int c)
{
int i;
int j;
i+=-a+b*2234e-2+i*(j+i/23.4)-100;
for(int i=j;i<100;i=i+1)
{
c+=2*(i+1);
if(i+j*8>233)
{
int i;
j=i;
}
else
{
i+=2;
}
}
i=i+1;
return i;
}
查看词法分析过程
查看语法分析过程
Dialog
语义分析结果
21: (j, _, _, dest_addr)
22: (@, a, _, T7)
23: (*, b, 2234e-2, T8)
24: (+, T7, T8, T9)
25: (/, i, 23.4, T10)
26: (+, j, T10, T11)
27: (*, i, T11, T12)
28: (+, T9, T12, T13)
29: (-, T13, 100, T14)
30: (+, i, T14, T15)
31: (:=, T15, _, i)
32: (:=, j, _, i)
33: (j<, i, 100, 36)
34: (:=, 0, _, T16)
35: (j, _, _, 37)
36: (:=, 1, _, T16)
37: (jnz, T16, _, 42)
38: (j, _, _, 59)
39: (+, i, 1, T17)
40: (:=, T17, _, i)
41: (j, _, _, 33)
42: (+, i, 1, T18)
43: (*, 2, T18, T19)
44: (+, c, T19, T20)
45: (:=, T20, _, c)
46: (*, j, 8, T21)
47: (+, i, T21, T22)
48: (j>, T22, 233, 51)
49: (:=, 0, _, T23)
50: (j, _, _, 52)
51: (:=, 1, _, T23)
52: (jnz, T23, _, 54)
53: (j, _, _, 56)
54: (:=, i, _, j)


MainWindow
选择文件
源程序
int program(int a,int b,int c)
{
int i;
int j;
i+=-a+b*2234e-2+i*(j+i/23.4)-100;
for(int i=j;i<100;i=i+1)
{
c+=2*(i+1);
if(i*j*8>233)
{
int i;
j=i;
}
else
{
i+=2;
}
}
i=i+1;
return i;
}
查看词法分析过程
查看语法分析过程
Dialog
语义分析结果
30: (+,i,T14,T15)
31: (:=,T15,_,i)
32: (:=,j,_,i)
33: (j<,i,100,36)
34: (:=,0,_,T16)
35: (j,_,_,37)
36: (:=,1,_,T16)
37: (jnz,T16,_,42)
38: (j,_,_,59)
39: (+,i,1,T17)
40: (:=,T17,_,i)
41: (j,_,_,33)
42: (+,i,1,T18)
43: (*,2,T18,T19)
44: (+,c,T19,T20)
45: (:=,T20,_,c)
46: (*,j,8,T21)
47: (+,i,T21,T22)
48: (j>,T22,233,51)
49: (:=,0,_,T23)
50: (j,_,_,52)
51: (:=,1,_,T23)
52: (jnz,T23,_,54)
53: (j,_,_,56)
54: (:=,i,_,j)
55: (j,_,_,58)
56: (+,i,2,T24)
57: (:=,T24,_,i)
58: (j,_,_,39)
59: (+,i,1,T25)
60: (:=,T25,_,i)
61: (:=,i,_,eax)
62: (j,_,_,dest_addr)

报错测试 1：


MainWindow
选择文件
源程序
编译分析
int fun(int i,int j
{
if(i<j*j+100/2)
{
i+=1;
while(i>j)
{
i=2;
j+=i;
}
}
return;
}
int program(int a,int b,int c)
{
int i;
int j;
i+=-a+b*2234e-2+i*(j+i/23.4)-100;
for(int i=j;i<100;i=i+1)
{
提示
ERROR!Please check around int{
OK
查看词法分析过程
查看语法分析过程
查看语义分析结果

报错测试 2：


MainWindow
选择文件
源程序
编译分析
int fun()
{
int i;
int j=20;
i=j+k;
return;
}
提示
There are variables(k) that are not defined!
OK
查看词法分析过程
查看语法分析过程
查看语义分析结果