编译原理实验报告

<https://github.com/Yuiffy>

2015年5月20日

目录

[**1.** **实验目的** 2](#_Toc419926740)

[**2.** **实验内容** 2](#_Toc419926741)

[**3.** **总体设计** 2](#_Toc419926742)

[**（1）** **符号表和关键字** 2](#_Toc419926743)

[符号表： 2](#_Toc419926744)

[关键字： 3](#_Toc419926745)

[**（2）** **文法产生式** 4](#_Toc419926746)

[**4.** **详细设计与实现** 5](#_Toc419926747)

[**5.** **总结** 6](#_Toc419926748)

[**6.** **测试样例** 6](#_Toc419926749)

[输入： 6](#_Toc419926750)

[词法分析输出： 8](#_Toc419926751)

[语法分析输出： 9](#_Toc419926752)

[语义分析输出： 26](#_Toc419926753)

[**7.** **代码** 26](#_Toc419926754)

[源文件： 26](#_Toc419926755)

[cf.h 26](#_Toc419926756)

[readTable.h 37](#_Toc419926757)

[yf.cpp 42](#_Toc419926758)

[输入文件： 48](#_Toc419926759)

[biao.txt 48](#_Toc419926760)

[source2.c 48](#_Toc419926761)

1. **实验目的**

实现C语言的词法分析器、语法分析器、语义分析与中间代码生成。

1. **实验内容**

实现了C语言词法分析器，能将.c文件读入，翻译成一个个符号和内容。

使用“编译工作台”软件将文法产生式生成LR（1）的ACTION表和GOTO表，用C语言程序将其读入，通过该程序能对词法分析的结果进行语法分析，判断是否能被该文法接收。

在语法分析过程中加入了语义分析，能将表达式转化为四地址码。

程序都是自己写的，对词法分析、语法分析、语义分析中间代码生成有了许多收获。

1. **总体设计**
2. **符号表和关键字**

### 符号表：

|  |  |
| --- | --- |
| letter(letter|digit)\* | 40 |
| digit digit\* | 41 |
| + | 42 |
| - | 43 |
| \* | 44 |
| / | 45 |
| % | 46 |
| ++ | 47 |
| -- | 48 |
| > | 49 |
| < | 50 |
| >= | 51 |
| <= | 52 |
| != | 53 |
| == | 54 |
| && | 55 |
| || | 56 |
| ! | 57 |
| & | 58 |
| | | 59 |
| ~ | 60 |
| ^ | 61 |
| << | 62 |
| >> | 63 |
| = | 64 |
| += | 65 |
| -= | 66 |
| \*= | 67 |
| /= | 68 |
| %= | 69 |
| &= | 70 |
| |= | 71 |
| ^= | 72 |
| >>= | 73 |
| <<= | 74 |
| , | 75 |
| ; | 76 |
| ( | 77 |
| ) | 78 |
| [ | 79 |
| ] | 80 |
| { | 81 |
| } | 82 |
| 浮点数digit\*.digit\* | 83 |
| . | 84 |
| ? | 85 |
| = | 86 |
| ” | 87 |
| ‘ | 88 |
| -> | 89 |

### 关键字：

|  |  |
| --- | --- |
| auto | 1 |
| double | 2 |
| int | 3 |
| struct | 4 |
| break | 5 |
| else | 6 |
| long | 7 |
| switch | 8 |
| case | 9 |
| enum | 10 |
| register | 11 |
| typedef | 12 |
| char | 13 |
| extern | 14 |
| return | 15 |
| union | 16 |
| const | 17 |
| float | 18 |
| short | 19 |
| unsigned | 20 |
| continue | 21 |
| for | 22 |
| signed | 23 |
| void | 24 |
| default | 25 |
| goto | 26 |
| sizeof | 27 |
| volatile | 28 |
| do | 29 |
| if | 30 |
| while | 31 |
| static | 32 |

1. **文法产生式**

**[非终结符]**

<程序> <函数定义> <类型> <变量> <参数声明> <函数块> <因式> <表达式> <变量> <因子> <运算符> <声明> <赋初值> <声明闭包> <声明语句闭包> <函数块闭包> <赋值函数> <循环for> <条件语句> <函数返回> <参数列表> <参数闭包> <参数> <否则语句>

**[终结符]**

{ } [ ] ( ) int id digit = + - < > == != \* / , string for if else return ；

**[文法起始符]**

<起始符>

**[生成式]**

<起始符> -> <程序>;

<程序> -> <函数定义> <程序> | <函数定义>;

<函数定义> -> <类型> <变量> ( <参数声明> ) { <函数块> };

<类型> -> int;

<变量> -> id [ <表达式> ] | id;

<因式> -> ( <表达式> ) | <变量> | digit ;

<表达式> -> <因子> <运算符> <表达式> | <因子>;

<因子> -> <因式> \* <因子> | <因式> / <因子> | <因式> ;

<运算符> -> + | - | < | > | == | !=;

<参数声明> -> <声明> <声明闭包> |;

<声明> -> <类型> <变量> <赋初值>;

<赋初值> -> = <表达式> |;

<声明闭包> -> , <声明> <声明闭包> |;

<函数块> -> <声明语句闭包> <函数块闭包>;

<声明语句闭包> -> <声明> ； <声明语句闭包> |;

<函数块闭包> -> <赋值函数> ； <函数块闭包> | <循环for> <函数块闭包> | <条件语句> <函数块闭包> | <函数返回> ； <函数块闭包> |;

<赋值函数> -> <变量> = <表达式> | <变量> ( <参数列表> ) ;

<参数列表> -> <参数> <参数闭包>;

<参数闭包> -> , <参数> <参数闭包> |;

<参数> -> id | digit | string;

<循环for> -> for ( <赋值函数> ； <表达式> ； <赋值函数> ) { <函数块闭包> };

<条件语句> -> if ( <表达式> ) { <函数块> } <否则语句>;

<否则语句> -> else { <函数块> } |;

<函数返回> -> return <表达式>;

1. **详细设计与实现**

**程序文件（共3个）：**

**cf.h**：词法分析的函数所在的头文件，内含vector<pair<int,string> > cf(string filename)，能将filename文件的内容读取成词法分析的结果（符号种别码与内容），存入可变长度数组vector中。

**readTable.h**：用于把编译工作台生成的action和goto表读入程序中。

**yf.cpp**：语法分析、语义分析的主函数，调用cf.h获得输入符号串，调用readTable.h获得action表和goto表。调用自身函数LR对符号串进行语法、语义分析，语法分析结果会输出到yfLog.txt中，语义分析结果会输出到控制台上。

int LR(vector<int> &w,vector<string> &wtk,int action[][MAX],int gto[][MAX], int A[],string B[], int beta[],const int &jing,string itos[MAX], FILE \*yufaFile)

参数说明：w为逆序的种别码输入序列，从后往前读。

wtk为内容输入序列，与种别码输入序列一一对应。

action[][]为action表

gto[][]为goto表

A[]为产生式的左半部，如r3: B->ab ,则A[2]=x，x为”B”的序号。

B[]为产生式的右部的字符串，便于输出检查用。

beta[]存着产生式右部的元素个数，用于得知规约时弹出多少个栈元素。

jing存着结束符#的序号。

itos[]存着每个字符编码所对应的字符，便于检查。

yufaFile存着语法分析结果要输出到哪个文件里。

**输入文件（共2个）：**

（将输入文件放到yf.cpp相同目录，才能使程序正常工作）

**source2.c**：词法分析器的输入文件，存着要分析的代码。

样例：

int main(){

int a=1;

int b;

int c;

int x;

b=2;

c=3;

x = a/b + b \* c \* (a+b);

if(a==1){

b=1;

}else{

b=2;

}

}

**biao.txt**：存着action表和goto表，以及一些手写的信息，便于程序读取两个表。

第一行含两个数字n，m。n为终结符的数量，m为非终结符的编号应该从多少开始。

第2到第n+1行，每行包含一个不含空格的字符串si和一个数xi，用空格隔开。表明这个终结符si在词法分析器中的编码为xi。

接下来是action表和goto表，第一行存着表头，也就是非终结符、终结符的名称。

第二行开始，第一个元素为行号，从0开始，之后是表的内容，每列之间用tab隔开。

（这里需要用chrome内核的浏览器打开编译工作台输出的htm格式的表，复制出来）

1. **总结**

通过实验，自己写了词法分析器、LR分析器、能将表达式翻译成四元式的语法制导文法分析器，实验中对课程上学习到的知识有了更深的了解，并且也知道了自己对哪些知识点了解的不深。

虽然没有实现从文法产生式生成LR(1)的action和goto表的程序、对控制类语句翻译进行中间代码生成，但也对其有了进一步的了解。

1. **测试样例**

## 输入：

source2.c：

int main(){

int a=1;

int b=2;

int c=3;

int x;

x = a/b + b \* c \* (a+b);

}

## 词法分析输出：

( 3 , int )

( 40 , main )

( 77 , ( )

( 78 , ) )

( 81 , { )

( 3 , int )

( 40 , a )

( 64 , = )

( 41 , 1 )

( 76 , ; )

( 3 , int )

( 40 , b )

( 64 , = )

( 41 , 2 )

( 76 , ; )

( 3 , int )

( 40 , c )

( 64 , = )

( 41 , 3 )

( 76 , ; )

( 3 , int )

( 40 , x )

( 76 , ; )

( 40 , x )

( 64 , = )

( 40 , a )

( 45 , / )

( 40 , b )

( 42 , + )

( 40 , b )

( 44 , \* )

( 40 , c )

( 44 , \* )

( 77 , ( )

( 40 , a )

( 42 , + )

( 40 , b )

( 78 , ) )

( 76 , ; )

( 82 , } )

## 语法分析输出：

1. action[0][3] = S4

stack: 0() 90($)

input: [int 3] [id 40] [( 77] [) 78] [{ 81] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

2. action[4][40] = r1

stack: 4() 3(int) 0() 90($)

input: [id 40] [( 77] [) 78] [{ 81] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=1) <类型>->int (goto[0][93]=3)(beta[1]=1)

3. action[3][40] = S7

stack: 3(int) 93(<类型>) 0() 90($)

input: [id 40] [( 77] [) 78] [{ 81] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

4. action[7][77] = r3

stack: 7() 40(id) 3(int) 93(<类型>) 0() 90($)

input: [( 77] [) 78] [{ 81] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[3][94]=6)(beta[3]=1)

5. action[6][77] = S8

stack: 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [( 77] [) 78] [{ 81] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

6. action[8][78] = r4

stack: 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [) 78] [{ 81] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=4) <参数声明>-> (goto[8][95]=11)(beta[4]=0)

7. action[11][78] = S22

stack: 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [) 78] [{ 81] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

8. action[22][81] = S46

stack: 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [{ 81] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

9. action[46][3] = S4

stack: 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

10. action[4][40] = r1

stack: 4() 3(int) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=1) <类型>->int (goto[46][93]=66)(beta[1]=1)

11. action[66][40] = S84

stack: 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

12. action[84][64] = r3

stack: 84() 40(id) 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[66][94]=83)(beta[3]=1)

13. action[83][64] = S104

stack: 83() 94(<变量>) 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

14. action[104][41] = S122

stack: 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

15. action[122][76] = r9

stack: 122() 41(digit) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=9) <因式>->digit (goto[104][97]=117)(beta[9]=1)

16. action[117][76] = r7

stack: 117() 97(<因式>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=7) <因子>-><因式> (goto[104][99]=119)(beta[7]=1)

17. action[119][76] = r8

stack: 119() 99(<因子>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=8) <表达式>-><因子> (goto[104][98]=124)(beta[8]=1)

18. action[124][76] = r25

stack: 124() 98(<表达式>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=25) <赋初值>->= <表达式> (goto[83][102]=103)(beta[25]=2)

19. action[103][76] = r19

stack: 103(<声明闭包>) 102(<赋初值>) 83() 94(<变量>) 66() 93(<类型>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=19) <声明>-><类型> <变量> <赋初值> (goto[46][101]=68)(beta[19]=3)

20. action[68][76] = S86

stack: 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

21. action[86][3] = S4

stack: 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

22. action[4][40] = r1

stack: 4() 3(int) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=1) <类型>->int (goto[86][93]=66)(beta[1]=1)

23. action[66][40] = S84

stack: 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

24. action[84][64] = r3

stack: 84() 40(id) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[66][94]=83)(beta[3]=1)

25. action[83][64] = S104

stack: 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [= 64] [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

26. action[104][41] = S122

stack: 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [digit 41] [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

27. action[122][76] = r9

stack: 122() 41(digit) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=9) <因式>->digit (goto[104][97]=117)(beta[9]=1)

28. action[117][76] = r7

stack: 117() 97(<因式>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=7) <因子>-><因式> (goto[104][99]=119)(beta[7]=1)

29. action[119][76] = r8

stack: 119() 99(<因子>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=8) <表达式>-><因子> (goto[104][98]=124)(beta[8]=1)

30. action[124][76] = r25

stack: 124() 98(<表达式>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=25) <赋初值>->= <表达式> (goto[83][102]=103)(beta[25]=2)

31. action[103][76] = r19

stack: 103(<声明闭包>) 102(<赋初值>) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=19) <声明>-><类型> <变量> <赋初值> (goto[86][101]=68)(beta[19]=3)

32. action[68][76] = S86

stack: 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

33. action[86][3] = S4

stack: 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [int 3] [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

34. action[4][40] = r1

stack: 4() 3(int) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=1) <类型>->int (goto[86][93]=66)(beta[1]=1)

35. action[66][40] = S84

stack: 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

36. action[84][64] = r3

stack: 84() 40(id) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[66][94]=83)(beta[3]=1)

37. action[83][64] = S104

stack: 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [= 64] [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

38. action[104][41] = S122

stack: 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [digit 41] [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

39. action[122][76] = r9

stack: 122() 41(digit) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=9) <因式>->digit (goto[104][97]=117)(beta[9]=1)

40. action[117][76] = r7

stack: 117() 97(<因式>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=7) <因子>-><因式> (goto[104][99]=119)(beta[7]=1)

41. action[119][76] = r8

stack: 119() 99(<因子>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=8) <表达式>-><因子> (goto[104][98]=124)(beta[8]=1)

42. action[124][76] = r25

stack: 124() 98(<表达式>) 104(<声明语句闭包>) 64(=) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=25) <赋初值>->= <表达式> (goto[83][102]=103)(beta[25]=2)

43. action[103][76] = r19

stack: 103(<声明闭包>) 102(<赋初值>) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=19) <声明>-><类型> <变量> <赋初值> (goto[86][101]=68)(beta[19]=3)

44. action[68][76] = S86

stack: 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

45. action[86][3] = S4

stack: 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [int 3] [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

46. action[4][40] = r1

stack: 4() 3(int) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=1) <类型>->int (goto[86][93]=66)(beta[1]=1)

47. action[66][40] = S84

stack: 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

48. action[84][76] = r3

stack: 84() 40(id) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[66][94]=83)(beta[3]=1)

49. action[83][76] = r10

stack: 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=10) <赋初值>-> (goto[83][102]=103)(beta[10]=0)

50. action[103][76] = r19

stack: 103(<声明闭包>) 102(<赋初值>) 83() 94(<变量>) 66() 93(<类型>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=19) <声明>-><类型> <变量> <赋初值> (goto[86][101]=68)(beta[19]=3)

51. action[68][76] = S86

stack: 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

52. action[86][40] = r20

stack: 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=20) <声明语句闭包>-> (goto[86][104]=106)(beta[20]=0)

53. action[106][40] = r30

stack: 106(<赋值函数>) 104(<声明语句闭包>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=30) <声明语句闭包>-><声明> ； <声明语句闭包> (goto[86][104]=106)(beta[30]=3)

54. action[106][40] = r30

stack: 106(<赋值函数>) 104(<声明语句闭包>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=30) <声明语句闭包>-><声明> ； <声明语句闭包> (goto[86][104]=106)(beta[30]=3)

55. action[106][40] = r30

stack: 106(<赋值函数>) 104(<声明语句闭包>) 86() 76(；) 68() 101(<声明>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=30) <声明语句闭包>-><声明> ； <声明语句闭包> (goto[86][104]=106)(beta[30]=3)

56. action[106][40] = r30

stack: 106(<赋值函数>) 104(<声明语句闭包>) 86() 76(；) 68() 101(<声明>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=30) <声明语句闭包>-><声明> ； <声明语句闭包> (goto[46][104]=69)(beta[30]=3)

57. action[69][40] = S93

stack: 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

58. action[93][64] = r3

stack: 93(<类型>) 40(id) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[69][94]=87)(beta[3]=1)

59. action[87][64] = S108

stack: 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [= 64] [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

60. action[108][40] = S121

stack: 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

61. action[121][45] = r3

stack: 121() 40(id) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[108][94]=116)(beta[3]=1)

62. action[116][45] = r6

stack: 116() 94(<变量>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=6) <因式>-><变量> (goto[108][97]=117)(beta[6]=1)

63. action[117][45] = S138

stack: 117() 97(<因式>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [/ 45] [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

64. action[138][40] = S121

stack: 138() 45(/) 117() 97(<因式>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

65. action[121][42] = r3

stack: 121() 40(id) 138() 45(/) 117() 97(<因式>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[138][94]=116)(beta[3]=1)

66. action[116][42] = r6

stack: 116() 94(<变量>) 138() 45(/) 117() 97(<因式>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=6) <因式>-><变量> (goto[138][97]=117)(beta[6]=1)

67. action[117][42] = r7

stack: 117() 97(<因式>) 138() 45(/) 117() 97(<因式>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=7) <因子>-><因式> (goto[138][99]=150)(beta[7]=1)

68. action[150][42] = r22

stack: 150() 99(<因子>) 138() 45(/) 117() 97(<因式>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=22) <因子>-><因式> / <因子> (goto[108][99]=119)(beta[22]=3)

69. action[119][42] = S29

stack: 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

70. action[29][40] = r13

stack: 29() 42(+) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=13) <运算符>->+ (goto[119][100]=139)(beta[13]=1)

71. action[139][40] = S121

stack: 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

72. action[121][44] = r3

stack: 121() 40(id) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[139][94]=116)(beta[3]=1)

73. action[116][44] = r6

stack: 116() 94(<变量>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=6) <因式>-><变量> (goto[139][97]=117)(beta[6]=1)

74. action[117][44] = S137

stack: 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [\* 44] [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

75. action[137][40] = S121

stack: 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

76. action[121][44] = r3

stack: 121() 40(id) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[137][94]=116)(beta[3]=1)

77. action[116][44] = r6

stack: 116() 94(<变量>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=6) <因式>-><变量> (goto[137][97]=117)(beta[6]=1)

78. action[117][44] = S137

stack: 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [\* 44] [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

79. action[137][77] = S120

stack: 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [( 77] [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

80. action[120][40] = S40

stack: 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

81. action[40][42] = r3

stack: 40(id) 40(id) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[120][94]=35)(beta[3]=1)

82. action[35][42] = r6

stack: 35() 94(<变量>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=6) <因式>-><变量> (goto[120][97]=36)(beta[6]=1)

83. action[36][42] = r7

stack: 36() 97(<因式>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

(k=7) <因子>-><因式> (goto[120][99]=38)(beta[7]=1)

84. action[38][42] = S29

stack: 38() 99(<因子>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [+ 42] [id 40] [) 78] [； 76] [} 82] [#]

85. action[29][40] = r13

stack: 29() 42(+) 38() 99(<因子>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [) 78] [； 76] [} 82] [#]

(k=13) <运算符>->+ (goto[38][100]=54)(beta[13]=1)

86. action[54][40] = S40

stack: 54(==) 100(<运算符>) 38() 99(<因子>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [id 40] [) 78] [； 76] [} 82] [#]

87. action[40][78] = r3

stack: 40(id) 40(id) 54(==) 100(<运算符>) 38() 99(<因子>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [) 78] [； 76] [} 82] [#]

(k=3) <变量>->id (goto[54][94]=35)(beta[3]=1)

88. action[35][78] = r6

stack: 35() 94(<变量>) 54(==) 100(<运算符>) 38() 99(<因子>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [) 78] [； 76] [} 82] [#]

(k=6) <因式>-><变量> (goto[54][97]=36)(beta[6]=1)

89. action[36][78] = r7

stack: 36() 97(<因式>) 54(==) 100(<运算符>) 38() 99(<因子>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [) 78] [； 76] [} 82] [#]

(k=7) <因子>-><因式> (goto[54][99]=38)(beta[7]=1)

90. action[38][78] = r8

stack: 38() 99(<因子>) 54(==) 100(<运算符>) 38() 99(<因子>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [) 78] [； 76] [} 82] [#]

(k=8) <表达式>-><因子> (goto[54][98]=73)(beta[8]=1)

91. action[73][78] = r23

stack: 73() 98(<表达式>) 54(==) 100(<运算符>) 38() 99(<因子>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [) 78] [； 76] [} 82] [#]

(k=23) <表达式>-><因子> <运算符> <表达式> (goto[120][98]=140)(beta[23]=3)

92. action[140][78] = S152

stack: 140() 98(<表达式>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [) 78] [； 76] [} 82] [#]

93. action[152][76] = r24

stack: 152() 78()) 140() 98(<表达式>) 120() 77(() 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [} 82] [#]

(k=24) <因式>->( <表达式> ) (goto[137][97]=117)(beta[24]=3)

94. action[117][76] = r7

stack: 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [} 82] [#]

(k=7) <因子>-><因式> (goto[137][99]=149)(beta[7]=1)

95. action[149][76] = r21

stack: 149() 99(<因子>) 137() 44(\*) 117() 97(<因式>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [} 82] [#]

(k=21) <因子>-><因式> \* <因子> (goto[137][99]=149)(beta[21]=3)

96. action[149][76] = r21

stack: 149() 99(<因子>) 137() 44(\*) 117() 97(<因式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [} 82] [#]

(k=21) <因子>-><因式> \* <因子> (goto[139][99]=119)(beta[21]=3)

97. action[119][76] = r8

stack: 119() 99(<因子>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [} 82] [#]

(k=8) <表达式>-><因子> (goto[139][98]=151)(beta[8]=1)

98. action[151][76] = r23

stack: 151() 98(<表达式>) 139() 100(<运算符>) 119() 99(<因子>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [} 82] [#]

(k=23) <表达式>-><因子> <运算符> <表达式> (goto[108][98]=131)(beta[23]=3)

99. action[131][76] = r38

stack: 131() 98(<表达式>) 108(<条件语句>) 64(=) 87(string) 94(<变量>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [} 82] [#]

(k=38) <赋值函数>-><变量> = <表达式> (goto[69][106]=89)(beta[38]=3)

100. action[89][76] = S109

stack: 89() 106(<赋值函数>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [； 76] [} 82] [#]

101. action[109][82] = r26

stack: 109(<函数返回>) 76(；) 89() 106(<赋值函数>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [} 82] [#]

(k=26) <函数块闭包>-> (goto[109][105]=132)(beta[26]=0)

102. action[132][82] = r39

stack: 132() 105(<函数块闭包>) 109(<函数返回>) 76(；) 89() 106(<赋值函数>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [} 82] [#]

(k=39) <函数块闭包>-><赋值函数> ； <函数块闭包> (goto[69][105]=88)(beta[39]=3)

103. action[88][82] = r29

stack: 88() 105(<函数块闭包>) 69() 104(<声明语句闭包>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [} 82] [#]

(k=29) <函数块>-><声明语句闭包> <函数块闭包> (goto[46][96]=67)(beta[29]=2)

104. action[67][82] = S85

stack: 67() 96(<函数块>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [} 82] [#]

105. action[85][90] = r28

stack: 85() 82(}) 67() 96(<函数块>) 46() 81({) 22(for) 78()) 11() 95(<参数声明>) 8() 77(() 6(else) 94(<变量>) 3(int) 93(<类型>) 0() 90($)

input: [#]

(k=28) <函数定义>-><类型> <变量> ( <参数声明> ) { <函数块> } (goto[0][92]=2)(beta[28]=8)

106. action[2][90] = r0

stack: 2() 92(<函数定义>) 0() 90($)

input: [#]

(k=0) <程序>-><函数定义> (goto[0][91]=1)(beta[0]=1)

107. action[1][90] = ACC

stack: 1() 91(<程序>) 0() 90($)

input: [#]

ACC

## 语义分析输出：

0: a=1

1: b=2

2: c=3

3: x=0

4: t0=a/b (0 = 1 / 2)

5: t1=a+b (3 = 1 + 2)

6: t2=c\*t1 (9 = 3 \* 3)

7: t3=b\*t2 (18 = 2 \* 9)

8: t4=t0+t3 (18 = 0 + 18)

9: x=t4

1. **代码**

## 源文件：

### cf.h

#include<stdio.h>

#include<string.h>

#include<iostream>

#include<cstring>

#include<map>

#include<vector>

using namespace std;

static const int TOKEN\_MAX = 1024;

static const int PROG\_MAX = 8192;

static char prog[PROG\_MAX];

static char token[TOKEN\_MAX];

static int p,len,m=0,row,sum=0;

static double dsum=0.0;

static map<string,int> tab;

void Init() {

tab.clear();

tab["auto"]=1;

tab["double"]=2;

tab["int"]=3;

tab["struct"]=4;

tab["break"]=5;

tab["else"]=6;

tab["long"]=7;

tab["switch"]=8;

tab["case"]=9;

tab["enum"]=10;

tab["registe"]=11;

tab["typedef"]=12;

tab["char"]=13;

tab["extern"]=14;

tab["return"]=15;

tab["union"]=16;

tab["const"]=17;

tab["float"]=18;

tab["short"]=19;

tab["unsigned"]=20;

tab["continue"]=21;

tab["for"]=22;

tab["signed"]=23;

tab["void"]=24;

tab["default"]=25;

tab["goto"]=26;

tab["sizeof"]=27;

tab["volatile"]=28;

tab["do"]=29;

tab["if"]=30;

tab["while"]=31;

tab["static"]=32;

}

int scaner() {

/\*

共分为三大块，分别是标示符、数字、符号，对应下面的 if else if 和 else

\*/

int syn = 0;

char ch;

memset(token,0,sizeof(token));

ch=prog[p++];

while(ch==' '|| ch=='\t' || ch=='\r') {

ch=prog[p];

p++;

}

if((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')) { //可能是标示符或者变量名

m=0;

while((ch>='0'&&ch<='9')||(ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')) {

token[m++]=ch;

ch=prog[p++];

}

token[m++]='\0';

p--;

string s = token;

syn = tab[s];

if(syn == 0)syn=40;

} else if((ch>='0'&&ch<='9') || (ch=='.')) { //数字

{

sum=0;

while((ch>='0'&&ch<='9')) {

sum=sum\*10+ch-'0';

ch=prog[p++];

}

}

double multi=0.1;

if(ch=='.') {

ch=prog[p++];

dsum=(double)sum;

while((ch>='0'&&ch<='9')) {

dsum += multi \* (ch-'0');

multi \*= 0.1;

ch=prog[p++];

}

p--;

syn=83;

} else {

p--;

syn=41;

}

} else switch(ch) { //其他字符

case'+':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='+') {

syn=47;

token[m++]=ch;

} else if(ch=='=') {

syn=65;

token[m++]=ch;

} else {

syn=42;

p--;

}

break;

case'-':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='-') {

syn=48;

token[m++]=ch;

} else if(ch=='=') {

syn=66;

token[m++]=ch;

} else if(ch=='>') {

syn=89;

token[m++]=ch;

} else {

syn=43;

p--;

}

break;

case'\*':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=') {

syn=67;

token[m++]=ch;

} else {

syn=44;

p--;

}

break;

case'/':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=') {

syn=68;

token[m++]=ch;

} else {

syn=45;

p--;

}

break;

case'%':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=') {

syn=69;

token[m++]=ch;

} else {

syn=46;

p--;

}

break;

case'<':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='<') {

token[m++]=ch;

ch=prog[p++];

if(ch=='=') {

syn=74;

token[m++]=ch;

} else {

syn=62;

p--;

}

} else if(ch=='=') {

syn=52;

token[m++]=ch;

} else {

syn=50;

p--;

}

break;

case'>':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='>') {

token[m++]=ch;

ch=prog[p++];

if(ch=='=') {

syn=73;

token[m++]=ch;

} else {

syn=63;

p--;

}

} else if(ch=='=') {

syn=51;

token[m++]=ch;

} else {

syn=49;

p--;

}

break;

case'!':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=') {

syn=53;

token[m++]=ch;

}

break;

case'=':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=') {

syn=54;

token[m++]=ch;

} else {

syn=64;

p--;

}

break;

case'&':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='&') {

syn=55;

token[m++]=ch;

} else if(ch=='=') {

syn=70;

token[m++]=ch;

} else {

syn=58;

p--;

}

break;

case'|':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='|') {

syn=56;

token[m++]=ch;

} else if(ch=='=') {

syn=71;

token[m++]=ch;

} else {

syn=59;

p--;

}

break;

case'~':

syn=60;

token[0]=ch;

break;

case'^':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=') {

syn=72;

token[m++]=ch;

} else {

syn=61;

p--;

}

break;

case',':

syn=75;

token[0]=ch;

break;

case';':

syn=76;

token[0]=ch;

break;

case'(':

syn=77;

token[0]=ch;

break;

case')':

syn=78;

token[0]=ch;

break;

case'[':

syn=79;

token[0]=ch;

break;

case']':

syn=80;

token[0]=ch;

break;

case'{':

syn=81;

token[0]=ch;

break;

case'}':

syn=82;

token[0]=ch;

break;

case'.':

syn=84;

token[0]=ch;

break;

case'?':

syn=85;

token[0]=ch;

break;

case':':

syn=86;

token[0]=ch;

break;

case'\"':

syn=87;

m=0;

token[m++]=ch;

ch=prog[p++];

while(ch!='"') {

token[m++]=ch;

ch=prog[p++];

if(ch=='"'&& prog[p-2]=='\\') {

token[m++]=ch;

ch=prog[p++];

}

if(p>=len) {

syn=-1;

break;

}

}

if(syn!=-1)token[m++]=ch;

break;

case'\'':

syn=87;

m=0;

token[m++]=ch;

ch=prog[p++];

while(ch!='\'') {

token[m++]=ch;

ch=prog[p++];

if(ch=='\''&& prog[p-2]=='\\') {

token[m++]=ch;

ch=prog[p++];

}

if(p>=len) {

syn=-1;

break;

}

}

if(syn!=-1)token[m++]=ch;

break;

case'\n':

syn=-2;

break;

default:

syn=-1;

token[0]=ch;

break;

}

return syn;

}

string itos(int x){

string re="";

if(x==0)return "0";

while(x){

re+='0'+(x%10);

x/=10;

}

for(int x=0;x<re.length()/2;x++){

re[x]=re[re.length()-1-x];

}

return re;

}

int stoi(string s){

int x=0;

int len=s.length();

int cf=1;

for(int i=len-1; i>=0; i--){

x+=(s[i]-'0')\*cf;

cf\*=10;

}

return x;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: cf

Description: 词法分析

Calls: Init,scaner

Input: 输入需要词法分析的源文件文件名

Output: 输出vector，存有词法分析得到的各个token

Return: token串

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

vector<pair<int,string> > cf(string filename) {

vector<pair<int,string> >re;

Init();

p=0;

row=1;

if(freopen(filename.data(),"r",stdin) == NULL) {

printf("Can\'t find the source code.\n");

return re;

}

char ch;

while((ch=getchar())!=EOF) {

prog[p++]=ch;

//printf("[%d,%c] ",ch,ch);

}

len = p;

p=0;

int syn;

do {

syn = scaner();

switch(syn) {

case 41:

cout<<"( "<<syn<<" , "<<sum<<" )"<<endl;

re.push\_back(make\_pair(syn,itos(sum)));

break;

case 83:

cout<<"( "<<syn<<" , "<<dsum<<" )"<<endl;

re.push\_back(make\_pair(syn,itos((int)dsum)));

break;

case -1:

printf("Error:Line %d, Found [%d,%c].\n",row,token[0],token[0]);

break;

case -2:

row++;

break;

default:

cout<<"( "<<syn<<" , "<<token<<" )"<<endl;

re.push\_back(make\_pair(syn,token));

break;

}

} while (p<len);

return re;

}

### readTable.h

#include<iostream>

#include<cstdio>

#include<map>

#include<cstring>

using namespace std;

static const int ACC=-10000;

static const int MAX=570;

static const int KO=-20000;

void Initag(int action[][MAX],int gto[][MAX]) {

int i,j;

for(i=0; i<MAX; i++)

for(j=0; j<MAX; j++) {

action[i][j]=KO;

gto[i][j]=KO;

}

}

void MakeShift(char c[],int action[][MAX],int& row,int& col,int column[MAX]) {

int i=6;

while(c[i]<'0'||c[i]>'9') {

if(c[i]=='\0') {

printf("MakeShiftError\n");

return;

}

i++;

}

int t=0;

while(c[i]>='0' && c[i]<='9') {

t=t\*10+c[i]-'0';

i++;

}

printf("action[%d][%d]=%d (col=%d)",row,column[col],t,col);

action[row][column[col]]=t;

puts(";");

}

static char c2[100][MAX];

void MakeReduce(char c[],int action[][MAX],int &row,int &col,int &rcnt,int A[], string B[], int beta[],map<string,int>&stoi, map<string,int> &rtable,int column[]) {

string red=c;

if(rtable.find(red)!=rtable.end()) {

action[row][column[col]]=-rtable[red]-1;///之前这里column[col]写成了col，怒错

printf("action[%d][%d]=%d , Already have :r%d\t",row,column[col],-rtable[red]-1,rtable[red]);

cout<<red<<endl;

return;

}

int ccnt,p;

int left;

string right;

int cr=0;

p=0;

ccnt=0;

int i,len;

len=strlen(c);

for(i=7; i<len; i++) {

if(c[i]!=' ') {

c2[ccnt][p++]=c[i];

} else {

if(p==0)continue;

c2[ccnt][p]='\0';

// printf("!%s,%d,%d!",c2[ccnt],i,len);

ccnt++;

p=0;

}

}

if(p!=0) {

c2[ccnt][p]='\0';

if(!(c2[ccnt][0]==-50 && c2[ccnt][1]==-75 && p==2)) {

//printf("!%d,%d,%d!",c2[ccnt][0],c2[ccnt][1],p);

ccnt++;

}

}

rtable[red]=rcnt;

action[row][column[col]]=-rcnt-1;

A[rcnt]=stoi[c2[0]];

B[rcnt]="";

for(i=2; i<ccnt; i++) {

B[rcnt]+=c2[i];

if(i!=ccnt-1)B[rcnt]+=" ";

}

beta[rcnt]=ccnt-2;

rcnt++;

// printf("reduce: ");

// for(i=0; i<ccnt;i++){

// printf("%s ",c2[i]);

// }

// puts("");

printf("action[%d][%d]=%d (col=%d)\t",row,column[col],action[row][column[col]],col);

cout<<"reduce: "<<A[rcnt-1]<<" -> "<<B[rcnt-1];

printf(" (beta[%d]=%d r%d)(",rcnt-1,beta[rcnt-1],rcnt-1);

cout<<red<<")\n";

return;

}

void MakeGto(char c[],int gto[][MAX],int& row,int &col, int column[MAX]) {

int i=0;

while(c[i]<'0'||c[i]>'9') {

if(c[i]=='\0') {

printf("MakeGtoError(%s)\n",c);

return;

}

i++;

}

int t=0;

while(c[i]>='0' && c[i]<='9') {

t=t\*10+c[i]-'0';

i++;

}

printf("gto[%d][%d]=%d (col=%d)",row,column[col],t,col);

gto[row][column[col]]=t;

puts(";");

}

void GetTable(const string& fname,int action[][MAX],int gto[][MAX], int A[],string B[], int beta[], int& jing, string itos[MAX]) {

if(freopen(fname.data(),"r",stdin) == NULL) {

printf("Can\'t find the source code.\n");

return;

}

printf("Log is in readTableLog.txt\n");

freopen("readTableLog.txt","w",stdout);

int cnt,cnt2;

string s;

int x;

map<string,int>stoi;

Initag(action,gto);

scanf("%d%d",&cnt,&cnt2);

for(int i=0; i<cnt; i++) {

cin>>s>>x;

stoi[s]=x;

itos[x]=s;

cout<<"itos["<<x<<"]="<<s<<endl;

}

jing=stoi["$"];

int column[MAX];

char c;

char cs[11111],cn=0,c1[11111];

int col=0, row=0;

getchar();

gets(cs);

int p=0;

int len=strlen(cs);

for(int i=0; i<len; i++) {

if(cs[i]!='\t') {

c1[p++]=cs[i];

} else {

c1[p]='\0';

string t=c1;

if (stoi.find(t)==stoi.end()) {

itos[cnt2]=t;

stoi[t]=cnt2++;

}

column[col++]=stoi[t];

printf("column[%d]=%d\t",col-1,column[col-1]);

cout<<t<<endl;

p=0;

}

}

///之前漏了下面这个，导致最后一列表头没读进去

if(p!=0) {

c1[p]='\0';

string t=c1;

if (stoi.find(t)==stoi.end()) {

itos[cnt2]=t;

stoi[t]=cnt2++;

}

column[col++]=stoi[t];

printf("column[%d]=%d\t",col-1,column[col-1]);

cout<<t<<endl;

p=0;

}

col=-1;

row=0;

int rcnt=0;

map<string,int> rtable;

while(scanf("%d",&row)!=EOF) {

gets(cs);

len=strlen(cs);

p=0;

col=-1;

for(int i=0; i<len; i++) {

if(cs[i]=='\t') {

c1[p]='\0';

if((p==1 && c1[0]!=' ') || p>1) {

if(c1[0]=='s') {

MakeShift(c1,action,row,col,column);

} else if(c1[0]=='r') {

MakeReduce(c1,action,row,col,rcnt,A,B,beta,stoi,rtable,column);

} else if(c1[0]=='a') {

action[row][column[col]]=ACC;

} else {

MakeGto(c1,gto,row,col,column);

}

}

p=0;

col++;

} else {

c1[p++]=cs[i];

}

}

///之前也少了下面那个，导致最后一列根本不读！

if(p>0) {

c1[p]='\0';

if((p==1 && c1[0]!=' ') || p>1) {

if(c1[0]=='s') {

MakeShift(c1,action,row,col,column);

} else if(c1[0]=='r') {

MakeReduce(c1,action,row,col,rcnt,A,B,beta,stoi,rtable,column);

} else if(c1[0]=='a') {

action[row][column[col]]=ACC;

} else {

MakeGto(c1,gto,row,col,column);

}

}

p=0;

col++;

}

}

freopen( "CON", "w", stdout );

printf("readTable Over\n");

return;

}

//int main() {

// int A[MAX];

// string B[MAX];

// int beta[MAX];

// int jing;

// int action[MAX][MAX];

// int gto[MAX][MAX];

// GetTable("biao.txt",action,gto,A,B,beta,jing);

// return 0;

//}

### yf.cpp

#include<iostream>

#include<cstdio>

#include<map>

#include<vector>

#include<vector>

#include "cf.h"

#include "readTable.h"

using namespace std;

void error(int x) {

if(x==0)printf("action ERROR\n");

else if (x==1)printf("goto ERROR\n");

else printf("Unknown Error\n");

}

void OutputCheck(vector<int>s,vector<int>w,int a,int act,int cas,int jing,string itos[], FILE\* yufaFile) {

fprintf(yufaFile, "%d.\taction[%d][%d] = ",cas, s.back(),a);

if(act>=0)fprintf(yufaFile, "S%d",act);

else if(act!=ACC && act!=KO)fprintf(yufaFile, "r%d",-act-1);

else if(act!=KO)fprintf(yufaFile, "ACC");

else fprintf(yufaFile, "NONE");

fprintf(yufaFile, "\nstack: ");

for(int i=s.size()-1; i>0; i--)

fprintf(yufaFile, "%d(%s) ",s[i],itos[s[i]].c\_str());

fprintf(yufaFile, "%d(%s)\n",s[0],itos[s[0]].c\_str());

fprintf(yufaFile, "input: ");

for(int i=w.size()-1; i>0; i--)

if(w[i]!=jing)fprintf(yufaFile, "[%s %d] ",itos[w[i]].c\_str(), w[i]);

else fprintf(yufaFile, "[#] ");

if(w[0]!=jing) fprintf(yufaFile, "[%s %d]",itos[w[0]].c\_str(), w[0]);

else fprintf(yufaFile, "[#]");

fprintf(yufaFile, "\n");

}

const int FUZHI=38; ///<赋值函数>-><变量> = <表达式>

const int BIAODASHI=23; ///<表达式>-><因子> <运算符> <表达式>

const int CHENG=21; ///<因子>-><因式> \* <因子>

const int CHU=22; ///<因子>-><因式> / <因子>

const int BIANLIANG=3; ///<变量>->id

const int DIGIT=9; ///<因式>->digit

const int BIAODA2=8; ///<表达式>-><因子>

const int YINZI=7; ///<因子>-><因式>

const int KUOHAO=24; ///<因式>->( <表达式> )

const int FUCHUZHI=25; ///<赋初值> -> = <表达式>

const int FUNULL=10; ///<赋初值> -> ε

const int SHENGMING=19; ///<声明> -> <类型> <变量> <赋初值>

const int IF=45; ///<条件语句>->if ( <表达式> ) { <函数块> } <否则语句>

const int ELSE=47; ///<否则语句>->else { <函数块> }

///从右往左，分别是2,4,6,....

pair<string,int> yuyi(int k,vector<pair<string,int> >stk,map<string,int> &mp,int A[], int &cnt, int &cnt2){

int len=stk.size();

if(k==BIANLIANG){ ///变量，从map里读值

string re=stk[len - 2].first;

int x=mp[re];

return make\_pair(re,x);

}else if(k==CHENG || k==CHU || k==BIAODASHI){

int t=cnt++;

string re="t"+itos(t);

int x=0 , y=stk[len-6].second, z=stk[len-2].second;

string op = stk[len - 4].first;

switch(k){

case CHENG:x=y\*z;break;

case CHU:

if(z==0)printf("ERROR:divide 0\n");

else x=y/z;

break;

case BIAODASHI:

if(op=="+")x=y+z;

else if(op=="-")x=y-z;

else if(op=="<")x=(y<z);

else if(op==">")x=(y>z);

else if(op=="==")x=(y==z);

else if(op=="!=")x=(y!=z);

else printf("Error(%s)\n",stk[len-4].first.c\_str());

}

printf("%d:\t",cnt2++);

cout<<re<<"="<<stk[len - 6].first<<stk[len - 4].first<<stk[len - 2].first;

printf(" \t(%d = %d %s %d)",x, y,op.c\_str(), z);

cout<<endl;

return make\_pair(re,x);

}else if(k==FUZHI){

string re=stk[len - 6].first;

int x=stk[len-2].second;

mp[re] = x;

printf("%d:\t",cnt2++);

cout<<stk[len - 6].first<<stk[len - 4].first<<stk[len - 2].first<<endl;

return make\_pair(re,x);

}else if(k==YINZI || k==BIAODA2){

return stk[len - 2];

}else if(k==DIGIT){

string s=stk[len-2].first;

int x=stoi(stk[len-2].first);

return make\_pair(s,x);

}else if(k==KUOHAO){

return stk[len-4];

}else if(k==SHENGMING){

string s=stk[len-4].first;

int x=stk[len-2].second;

mp[s]=x;

printf("%d:\t",cnt2++);

cout<<s<<"="<<x<<endl;

return make\_pair(s,x);

}else if(k==FUNULL){

return make\_pair("",0);

}else if(k==IF){

}else return stk[len - 2];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: LR

Description: 语法分析，判断该语言能否被接受

Calls: error,OutputCheck

Input: w为逆序的输入数字序列，vector类型，可用pop\_back删除最后一个/back取得最后一个的值

action[][]为action表，action[x][y]=z表示栈顶为x，输入为a时的操作，大于等于0为移进并跳转到状态Sz

小于0且大于ACC表示按(-z-1)号产生式进行归约

等于ACC（设为-10000)表示可接受

gto[][]为goto表，归并弹出2\*beta个符号后要把生成式左端A和gto[S'][A]先后压入栈中，S'为栈顶状态

A为产生式表左端，B为产生式表右端，beta为产生式 表右端元素数量，jing为#号的编号

Output: 输出语法分析过程，栈/输入缓冲区

Return: 返回0说明ACC

Others: // 其它说明

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int LR(vector<int> &w,vector<string> &wtk,int action[][MAX],int gto[][MAX], int A[],string B[], int beta[],const int &jing,string itos[MAX], FILE \*yufaFile) {

vector<int>s;

vector<pair<string,int> >stk;

map<string,int> mp;

int idcnt=0;

int sencnt=0;

s.clear();

stk.clear();

s.push\_back(jing);

s.push\_back(0);

stk.push\_back(make\_pair("#",0));

stk.push\_back(make\_pair("0",0));

int cas=1;

while(true) {

int a=w.back();

string b=wtk.back();

//printf("%d,%d\n",s.back(),a);

int act=action[s.back()][a];

OutputCheck(s,w,a,act,cas++,jing,itos, yufaFile);

if(act>=0) { ///移进

s.push\_back(a);

s.push\_back(act);

w.pop\_back();

stk.push\_back(make\_pair(b,0));

stk.push\_back(make\_pair("act",0));

wtk.pop\_back();

} else if (act > ACC) { ///规约

int k= -act - 1;

pair<string,int> id = yuyi(k,stk,mp,A,idcnt,sencnt);

for(int i=0; i<2\*beta[k]; i++){

s.pop\_back();

stk.pop\_back();

}

int s\_=s.back();

s.push\_back(A[k]);

stk.push\_back(id);

if(gto[s\_][A[k]]==KO) {

fprintf(yufaFile, "gto[%d][%d]=KO , beta[%d]=%d\n",s\_,A[k],k,beta[k]);

error(1);

break;

}

s.push\_back(gto[s\_][A[k]]);

stk.push\_back(make\_pair("gto",0));

fprintf(yufaFile, "(k=%d)\t%s->%s",k,itos[A[k]].c\_str(), B[k].c\_str());

fprintf(yufaFile, "\t (goto[%d][%d]=%d)(beta[%d]=%d)\n",s\_,A[k],gto[s\_][A[k]],k,beta[k]);

} else if (act==ACC) return 0;///完成

else {///错误

error(0);

break;

}

fprintf(yufaFile, "\n");

}

}

int action[MAX][MAX]= {{3,4,KO},{KO,KO,ACC},{3,4,KO},{3,4,KO},{-3,-3,-3},{-1,-1,-1},{-2,-2,-2}};

int gto[MAX][MAX]= {{KO,KO,KO,1,2},

{KO,KO,KO,KO,KO},

{KO,KO,KO,KO,5},

{KO,KO,KO,KO,6},

{KO,KO,KO,KO,KO},

{KO,KO,KO,KO,KO},

{KO,KO,KO,KO,KO}

};

int main() {

int A[MAX]= {3,4,4};

string B[MAX]= {"BB","aB","b"};

string itos[MAX];

///a=0,b=1,#=2,S=3,B=4

int beta[MAX]= {2,2,1};

int jing=2;

vector<int> w;

vector<string> wtk;

vector<pair<int,string> > inp;

GetTable("biao.txt",action,gto,A,B,beta,jing,itos);

inp=cf("source2.c");

inp.push\_back(make\_pair(jing,"#"));

for(int i=inp.size()-1; i>=0; i--){

w.push\_back(inp[i].first);

wtk.push\_back(inp[i].second);

}

FILE\* yufaFile = fopen("yfLog.txt","w");

printf("===================\n");

if(LR(w,wtk,action,gto,A,B,beta,jing,itos,yufaFile)==0) fprintf(yufaFile, "ACC");

else fprintf(yufaFile, "ERROR");

return 0;

}

## 输入文件：

### biao.txt



### source2.c

int main(){

int a=1;

int b;

int c;

int x;

b=2;

c=3;

x = a/b + b \* c \* (a+b);

if(a==1){

b=1;

}else{

b=2;

}

}