提交日期：2016年10月24日

实验二

**CMM 语言总体架构的设计和搭建**

1. 【实验环境】（使用的软件）：

IDEA 2016.1.3

JDK1.8

Antlr4.5.3

1. 【参考资料】：

《编译原理》第二版

《编译原理及实践》

1. 系统的概要设计

**系统概述：**实现CMM语法功能的解释器。

**系统构架（暂时）：**

1. **系统分为四个子模块：前台界面模块，词法分析模块，语法分析模块，语义分析模块。**

前台界面模块：为用户提供编辑，输入，输出功能。

词法分析模块：将用户的输入识别为单个的Token，并且显示出来，传给语法分析部分使用。

语法分析模块：接受词法分析模块的Token流，生成AST树，检查用户输入的文本是否符合文法规则，并且提供报错。

语义分析模块：深度遍历AST树的每个节点，并解释含义执行。检查输入文本语义是否正确，并且提供报错。

各个模块之间相互影响，并且功能独立，语法分析其实已经嵌套了词法分析的功能，词法分析模块只是起到显示的作用。

**2． 模块之间的关系：**用户在前台界面模块输入内容。词法分析模块分析输入的结果，语法分析模块也分析输入结果，同时语法分析生成AST树，语义分析模块对AST树进行语义分析。

**CMM 语言词法分析**

1. 【实验环境】（使用的软件）：

IDEA 2016.1.3

JDK1.8

Antlr4.5.3

1. 【参考资料】：

《编译原理》第二版

《编译原理及实践》

网络资料：

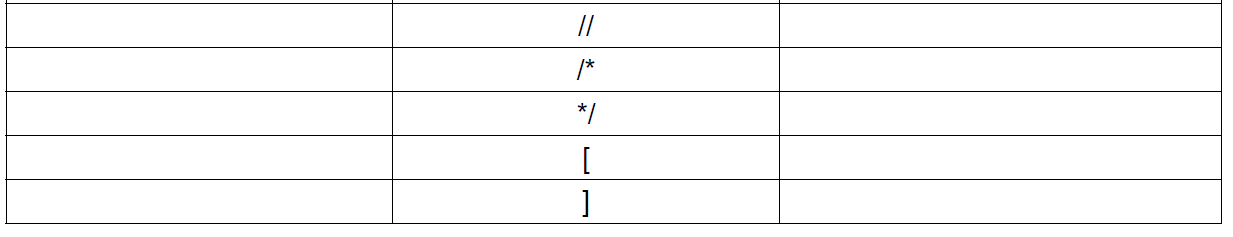
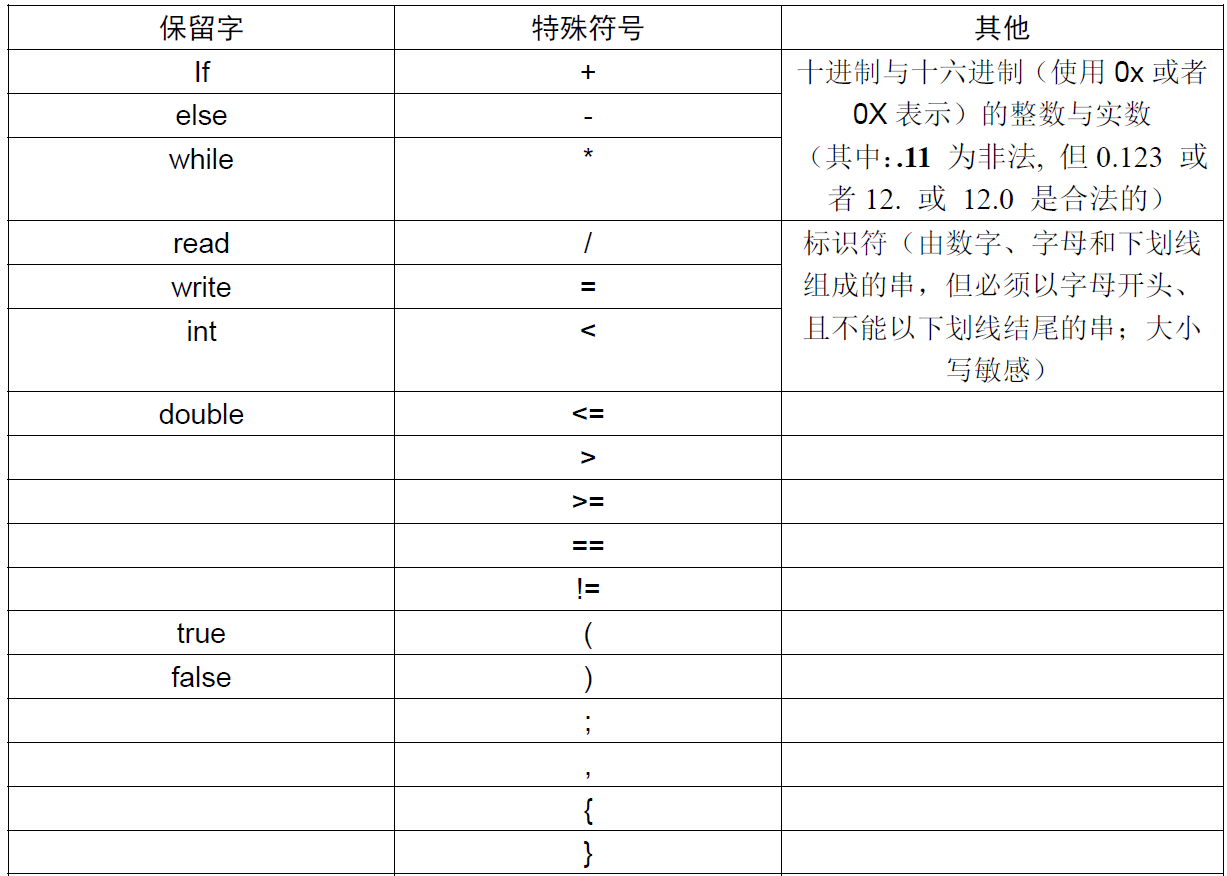
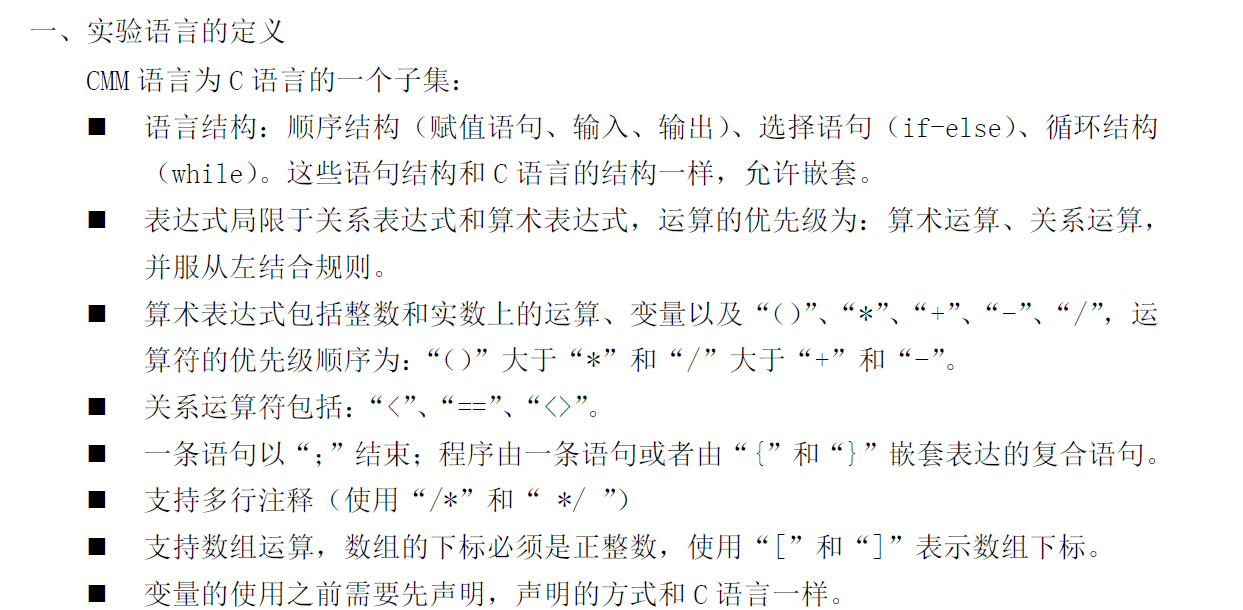
http://www.antlr.org/api/Java/index.html

http://www.tuicool.com/articles/JJ7j2i

http://www.cnblogs.com/6DAN\_HUST/archive/2008/12/26/1363011.html

http://www.cnblogs.com/lifesting/archive/2012/07/17/2595536.html

1. 词法分析



**1.表示符：**

由数字，字母，下划线组成，开头不能为数字。

**2.基本数据类型：**

整形( int ) 浮点(double)

**3.变量和常量:**

**3.1.变量**：

3.1.1变量定义：

int i ,j ;

double b ;

**3.2.文字常量**：

3.2.1. 整形符号常量：

15 //十进制

0x21 //十六进制

3.2.2.实行常量：

0.12 ，1.23 //一般形式

1.20 ，12.

**4.运算符和表达式**

**运算符及优先级和结合**

|  |  |  |
| --- | --- | --- |
| **运算符** | **解释** | **结合方式** |
| **() [] .** | **括号（函数等），数组，结构成员访问** | **由左向右** |
| **+ -** | **正负号** | **由右向左** |
| **\* /** | **乘，除** | **由左向右** |
| **+ -** | **加，减** | **由左向右** |
| **<= >= >** | **小于，小于等于，大于等于，大于** | **由左向右** |
| **== !=** | **等于，不等于** | **由左向右** |
| **=** | **各种赋值** | **由右向左** |
| **,** | **逗号（顺序）** | **由左向右** |

**5.数组：**

**5.1.数组的定义**：

一维数组：数据类型 数组名 [常量] ;

二维数组：数组类型 数组名 [常量][常量] ;

**5.2.数组的使用：**

一维数组：数组名[表达式]

二维数组：数组名[表达式][表达式]

**6.语句**

**6.1.表达式语句：**

形式： 表达式 ;

类别： 赋值语句，行注释， 文本注释

**6.2.复合语句 ：**

由”{ }”括起来的一组语句构成的一个复合语句

**6.3. 流程控制语句**（**支持嵌套**）

6**.3.1 if语句：**

if(表达式) 语句 1 ;

《else if 语句 2 ;…》

《else 语句 3 》

* + 1. **while语句**

while（表达式）循环体语句

* + 1. **for循环语句**

for（表达式1 ；表达式2 ；表达式3）循环体语句 //表达式1必须为申明

for（表达式1 ；表达式2 ； ）循环体语句

for（ ；表达式2 ； ）循环体语句

for（ ；表达式2 ；表达式3）循环体语句

**6.4.输入输出语句**

输入： 变量名 =read( ) ;

输出： write(“ 提示信息” ，表达式) ;

**6.5.函数定义语句**

**返回类型 函数名（参数列表）{ 语句}**

\*返回类型：void int char real

\*参数列表：《（基本数据类型 ,标示符）…》

\*语句不可以为空

**6.6函数调用语句：**

**函数名（参数）;**

参数可以是类型一致的表达式，也可以为常量

**7.注释**

文法中的COMMENT、LINE\_COMMENT和WS规则只是为了滤掉相应内容，没有必要与语法规则关联，这样它们也可以正确的工作。COMMENT符号使用了“.”符号来匹配一切字符直到匹配“/”字符为止。LINE\_COMMENT从“//”字符开始匹配除了“\r\n”以外的所有字符直到匹配“\r\n”为止。其中有可以没有回车符只有换行符所以“\r”是可选项。

文法如下，（已生成java代码）

**grammar** CMMLEXER;  
  
*//RESERVE WORD*IF : **'if'**;  
ELSE : **'else'**;  
WHILE : **'while'**;  
READ : **'read'**;  
WRITE : **'write'**;  
INT : **'int'**;  
DOUBLE : **'double'**;  
TRUE : **'true'**|**'TRUE'**;  
FALSE : **'false'**|**'FALSE'**;  
  
*//OPERATOR*PLUS : **'+'**;  
MINUS : **'-'**;  
MULTIPLICATION : **'\*'**;  
DIVISION : **'/'**;  
MOD : **'%'** ;  
ASSIGNMENT : **'='**;  
LESSTHAN : **'<'**;  
LESS\_EQUAL : **'<='**;  
MORETHAN : **'>'**;  
MORE\_EQUAL : **'>='**;  
EQUAL : **'=='**;  
UNEQUAL : **'!='**;  
  
*//DELIMITER*LEFT\_LITTLE\_BRACE : **'('**;  
RIGHT\_LITTLE\_BRACE : **')'**;  
SEMICOLONE : **';'**;  
COMMA : **','**;  
LEFT\_BRACE : **'{'**;  
RIGHT\_BRACE : **'}'**;  
  
LEFT\_ARRAY\_BRACE : **'['**;  
RIGHT\_ARRAY\_BRACE : **']'**;  
  
*//IDENTIFIER*ID : LETTER(LETTER|DecDigit|**'\_'**)\*(LETTER|DecDigit) ;  
  
WS : [ \t\n\r]+ -> skip ;  
LINE\_COMMENT : **'//'** ~[\r\n]\* **'\r'**? **'\n'** ;  
COMMENT : **'/\*'** .\*? **'\*/'** ;  
  
*//CONSTANT***fragment**LETTER  
 : [A-Z]  
 | [a-z]  
 ;  
DECIMAL : (**'0'** | [1-9](DecDigit)\*) ;  
HEX : **'0'** (**'x'**|**'X'**) HexDigit+ ;  
**fragment**DecDigit : [0-9];  
**fragment**HexDigit : ([0-9]|[a-f]|[A-F]) ;  
  
FLOATING  
 : DECIMAL **'.'** (DecDigit)\*  
 | (**'0'**..**'9'**)+ **'.'** | HEX **'.'** HexDigit\*  
 ;

为了将每个词素分类，我在Lexer.java文件中添加了如下代码：

*//新声明TAG来标识token，为在main中使用，将以下数组设为公有***public static final** String[] ***\_TOKEN\_TAGS*** = {  
 **null**, **"reserved word:"**, **"reserved word:"**, **"reserved word:"**, **"reserved word:"**,  
 **"reserved word:"**, **"reserved word:"**, **"reserved word:"**, **"reserved word:"**,  
 **"reserved word:"**, **"operator:"**, **"operator:"**, **"operator:"**, **"operator:"**, **"operator:"**,  
 **"operator:"**, **"operator:"**, **"operator:"**, **"operator:"**, **"operator:"**, **"operator:"**,  
 **"operator:"**, **"delimiter:"**, **"delimiter:"**, **"delimiter:"**, **"delimiter:"**, **"delimiter:"**,  
 **"delimiter:"**, **"delimiter:"**, **"delimiter:"**, **"ID, name="**, **"WS"**, **"LINE\_COMMENT"**,  
 **"COMMENT"**, **"NUM, val="**, **"NUM, val="**, **"NUM, val="**};

数组中每个元素与之前token一一对应，token名的数组如下：

**private static final** String[] ***\_SYMBOLIC\_NAMES*** = {  
 **null**, **"IF"**, **"ELSE"**, **"WHILE"**, **"READ"**, **"WRITE"**, **"INT"**, **"DOUBLE"**, **"TRUE"**,   
 **"FALSE"**, **"PLUS"**, **"MINUS"**, **"MULTIPLICATION"**, **"DIVISION"**, **"MOD"**, **"ASSIGNMENT"**,   
 **"LESSTHAN"**, **"LESS\_EQUAL"**, **"MORETHAN"**, **"MORE\_EQUAL"**, **"EQUAL"**, **"UNEQUAL"**,   
 **"LEFT\_LITTLE\_BRACE"**, **"RIGHT\_LITTLE\_BRACE"**, **"SEMICOLONE"**, **"COMMA"**, **"LEFT\_BRACE"**,   
 **"RIGHT\_BRACE"**, **"LEFT\_ARRAY\_BRACE"**, **"RIGHT\_ARRAY\_BRACE"**, **"ID"**, **"WS"**, **"LINE\_COMMENT"**,   
 **"COMMENT"**, **"DECIMAL"**, **"HEX"**, **"FLOATING"**};

在测试过程中，发现了最后一行的注释不能正常显示的问题，经过查阅资料，在文法的COMMENT定义中，原定义：，现将定义改为：



然后在测试中还发现了ID（字母开头含字母数字下划线结尾为字母数字）的匹配问题，发现是忽略了ID为单个字母的情况，修改为：

然后对java代码进行了多次修改以满足测试要求，最终代码为：

//main.java

**import** org.antlr.v4.runtime.\*;  
  
**import** java.io.\*;  
**import** java.util.ArrayDeque;  
  
**public class** Main {  
  
 **public static void** main(String[] args) **throws** IOException {  
 *// create a CharStream that reads from file* String fileName = **"test1.cmm"**;  
 File file = **new** File(fileName);  
 FileInputStream inputFile = **new** FileInputStream(file);  
 LineNumberReader lineNumberReader = **new** LineNumberReader(**new** FileReader(file)); *//文件流操作参考自官方文档  
 //lexer file in* ANTLRInputStream input = **new** ANTLRInputStream(inputFile);  
 CMMLEXERLexer lexer = **new** CMMLEXERLexer(input);  
 *//队列存储token* ArrayDeque<Token> arrayDeque = **new** ArrayDeque<>();  
 String[] tokenTags = CMMLEXERLexer.***\_TOKEN\_TAGS***;  
 **int** lineNum = 1;  
 *//visit整个文件并在eof的时候停止* **for** (Token token = lexer.nextToken(); token.getType() != Token.***EOF***; token = lexer.nextToken()) {  
 *//转到下一行第一个token的时候开始处理上一行的输出，若多行注释则在遍历到注释后第一个token时整体输出* **if** (lineNum != token.getLine()) {  
 *//整行先输出* **for** (**int** i=lineNum; i<token.getLine(); i++) {  
 lineNumberReader.setLineNumber(i);  
 String s = lineNumberReader.readLine();  
 System.***out***.printf(**"%d: %s\n"**, i, s);  
 }  
 *//非注释行逐个token输出* **while** (!arrayDeque.isEmpty()) {  
 Token t = arrayDeque.pollFirst();  
 **if** (!tokenTags[t.getType()].equals(**"COMMENT"**) && !tokenTags[t.getType()].equals(**"LINE\_COMMENT"**))  
 {  
 System.***out***.printf(**"\t%d: %s %s\n"**, lineNum, tokenTags[t.getType()], t.getText());  
 }  
 }  
 *//进入下一行* lineNum = token.getLine();  
 }  
 arrayDeque.addLast(token); *//同一行轮流进queue* }  
 *//EOF前一行* Token token = lexer.nextToken();  
 **if** (token.getType() == Token.***EOF***){  
 lineNumberReader.setLineNumber(lineNum);  
 String s = lineNumberReader.readLine();  
 System.***out***.printf(**"%d: %s\n"**, lineNum, s);  
 lineNum++;  
 System.***out***.println(**"\t"**+lineNum+**": EOF"** );  
 }  
 inputFile.close();  
 }  
}

注：本程序生成的\*.jar文件是以测试用例test1.cmm为模版生成的，所以运行的输出应为test1.cmm的测试输出

1. 测试：

1、

//erroe1\_ID.cmm

/\*\*  
 \* ��־������ ����  
 \* 2010-11-13  
 \* ������  
 \*   
 \* ���Ա��������Ƿ�Ϸ� ��Ҫ�ǣ�  
 \* 1.��ʶ���Ƿ����� �����淶(����ĸ��  
 \* ���֡��»�����ɣ�ֻ������ĸ��ͷ  
 \* ���Ҳ������»��߽���)  
 \* 2.����ͬ�������ڲ��������ͬ������  
 \*/  
  
  
//���Ա�ʶ���Ƿ�Ϸ�  
int \_a;  
real 3\_a; //���ֿ�ͷ  
int a\_; //�»��߽�β  
int a@com; //�Ƿ��ַ�  
  
int i=0;  
while(i<6)  
{  
 write(i);  
 i=i+1;  
}  
  
//��ͬ�������ڶ���ͬ������  
int i=10;  
  
/\*output  
����������Ϣ���ܸ���  
\*///:~

输出：

1: /\*\*

2: \* ��־������ ����

3: \* 2010-11-13

4: \* ������

5: \*

6: \* ���Ա��������Ƿ�Ϸ� ��Ҫ�ǣ�

7: \* 1.��ʶ���Ƿ����� �����淶(����ĸ��

8: \* ���֡��»�����ɣ�ֻ������ĸ��ͷ

9: \* ���Ҳ������»��߽���)

10: \* 2.����ͬ�������ڲ��������ͬ������

11: \*/

12:

13:

14: //���Ա�ʶ���Ƿ�Ϸ�

line 15:4 token recognition error at: '\_'

line 16:6 token recognition error at: '\_'

line 17:5 token recognition error at: '\_'

line 18:5 token recognition error at: '@'

15: int \_a;

15: reserved word: int

15: ID, name= a

15: delimiter: ;

16: real 3\_a; //���ֿ�ͷ

16: ID, name= real

16: NUM, val= 3

16: ID, name= a

16: delimiter: ;

17: int a\_; //�»��߽�β

17: reserved word: int

17: ID, name= a

17: delimiter: ;

18: int a@com; //�Ƿ��ַ�

19:

18: reserved word: int

18: ID, name= a

18: ID, name= com

18: delimiter: ;

20: int i=0;

20: reserved word: int

20: ID, name= i

20: operator: =

20: NUM, val= 0

20: delimiter: ;

21: while(i<6)

21: reserved word: while

21: delimiter: (

21: ID, name= i

21: operator: <

21: NUM, val= 6

21: delimiter: )

22: {

22: delimiter: {

23: write(i);

23: reserved word: write

23: delimiter: (

23: ID, name= i

23: delimiter: )

23: delimiter: ;

24: i=i+1;

24: ID, name= i

24: operator: =

24: ID, name= i

24: operator: +

24: NUM, val= 1

24: delimiter: ;

25: }

26:

25: delimiter: }

27: //��ͬ�������ڶ���ͬ������

28: int i=10;

29:

28: reserved word: int

28: ID, name= i

28: operator: =

28: NUM, val= 10

28: delimiter: ;

30: /\*output

31: ����������Ϣ���ܸ���

32: \*///:~

33: EOF

Process finished with exit code 0

2、

//**error3\_comment.cmm**

/\*\*  
 \* ע�ͱ��� ����  
 \* 2010-11-13  
 \* ������  
 \*   
 \* ע�ͱ��� ���� ��Ҫ�ǣ�  
 \* 1.ע��Ƕ��  
 \* 2.����ע���޽�β  
 \*/  
  
int I[6];  
  
int i=0;  
while(i<6)  
{  
 I[i] = i;  
 i=i+1;  
}  
  
/\*output:  
����������Ϣ���ܸ���  
\*///:~  
  
//ע��Ƕ��  
/\* ffff /\* mmmm \*/ fffff \*/  
//����ע���޽�β  
/\* ����...//

输出：

1: /\*\*

2: \* ע�ͱ��� ����

3: \* 2010-11-13

4: \* ������

5: \*

6: \* ע�ͱ��� ���� ��Ҫ�ǣ�

7: \* 1.ע��Ƕ��

8: \* 2.����ע���޽�β

9: \*/

10:

11: int I[6];

12:

11: reserved word: int

11: ID, name= I

11: delimiter: [

11: NUM, val= 6

11: delimiter: ]

11: delimiter: ;

13: int i=0;

13: reserved word: int

13: ID, name= i

13: operator: =

line 27:3 token recognition error at: '�'

line 27:4 token recognition error at: '�'

line 27:5 token recognition error at: '�'

line 27:6 token recognition error at: '�'

line 27:7 token recognition error at: '.'

line 27:8 token recognition error at: '.'

line 27:9 token recognition error at: '.'

13: NUM, val= 0

13: delimiter: ;

14: while(i<6)

14: reserved word: while

14: delimiter: (

14: ID, name= i

14: operator: <

14: NUM, val= 6

14: delimiter: )

15: {

15: delimiter: {

16: I[i] = i;

16: ID, name= I

16: delimiter: [

16: ID, name= i

16: delimiter: ]

16: operator: =

16: ID, name= i

16: delimiter: ;

17: i=i+1;

17: ID, name= i

17: operator: =

17: ID, name= i

17: operator: +

17: NUM, val= 1

17: delimiter: ;

18: }

19:

18: delimiter: }

20: /\*output:

21: ����������Ϣ���ܸ���

22: \*///:~

23:

24: //ע��Ƕ��

25: /\* ffff /\* mmmm \*/ fffff \*/

25: ID, name= fffff

25: operator: \*

25: operator: /

26: //����ע���޽�β

27: /\* ����...//

28: EOF

Process finished with exit code 0

3、

//**test1.cmm**

/\*\*  
 \* ���� ��������  
 \* 2010-11-13  
 \* ������  
 \*   
 \* ���Ա��������Ƿ�Ϸ� ��Ҫ�ǣ�  
 \* 1.����ͬ�������ڲ��������ͬ������  
 \* 2.��ͬ�������������ϼ�ͬ������  
 \*/  
  
//���Ա�ʶ���Ƿ�Ϸ�  
int a\_2;  
double r\_2\_r;  
double i,j=23,k;  
  
  
//���Բ�ͬ��������ͬ����������  
int c = 23;  
double b[2];  
b[0] = 23;  
b[1] = 0;  
if(b[1] == 0)  
{  
 double c = 23.5;  
 if(c == 23.5)  
 write(1);  
 else  
 write(0);  
  
 double b[1];  
 b[0] = 0.05;  
 if(b[0] == 0.05)  
 write(1);  
 else  
 write(0);  
}  
/\*output:  
1  
1  
\*/  
//:~

输出：

1: /\*\*

2: \* ���� ��������

3: \* 2010-11-13

4: \* ������

5: \*

6: \* ���Ա��������Ƿ�Ϸ� ��Ҫ�ǣ�

7: \* 1.����ͬ�������ڲ��������ͬ������

8: \* 2.��ͬ�������������ϼ�ͬ������

9: \*/

10:

11: //���Ա�ʶ���Ƿ�Ϸ�

12: int a\_2;

12: reserved word: int

12: ID, name= a\_2

12: delimiter: ;

13: double r\_2\_r;

13: reserved word: double

13: ID, name= r\_2\_r

13: delimiter: ;

14: double i,j=23,k;

15:

16:

14: reserved word: double

14: ID, name= i

14: delimiter: ,

14: ID, name= j

14: operator: =

14: NUM, val= 23

14: delimiter: ,

14: ID, name= k

14: delimiter: ;

17: //���Բ�ͬ��������ͬ����������

18: int c = 23;

18: reserved word: int

18: ID, name= c

18: operator: =

18: NUM, val= 23

18: delimiter: ;

19: double b[2];

19: reserved word: double

19: ID, name= b

19: delimiter: [

19: NUM, val= 2

19: delimiter: ]

19: delimiter: ;

20: b[0] = 23;

20: ID, name= b

20: delimiter: [

20: NUM, val= 0

20: delimiter: ]

20: operator: =

20: NUM, val= 23

20: delimiter: ;

21: b[1] = 0;

21: ID, name= b

21: delimiter: [

21: NUM, val= 1

21: delimiter: ]

21: operator: =

21: NUM, val= 0

21: delimiter: ;

22: if(b[1] == 0)

22: reserved word: if

22: delimiter: (

22: ID, name= b

22: delimiter: [

22: NUM, val= 1

22: delimiter: ]

22: operator: ==

22: NUM, val= 0

22: delimiter: )

23: {

23: delimiter: {

24: double c = 23.5;

24: reserved word: double

24: ID, name= c

24: operator: =

24: NUM, val= 23.5

24: delimiter: ;

25: if(c == 23.5)

25: reserved word: if

25: delimiter: (

25: ID, name= c

25: operator: ==

25: NUM, val= 23.5

25: delimiter: )

26: write(1);

26: reserved word: write

26: delimiter: (

26: NUM, val= 1

26: delimiter: )

26: delimiter: ;

27: else

27: reserved word: else

28: write(0);

29:

28: reserved word: write

28: delimiter: (

28: NUM, val= 0

28: delimiter: )

28: delimiter: ;

30: double b[1];

30: reserved word: double

30: ID, name= b

30: delimiter: [

30: NUM, val= 1

30: delimiter: ]

30: delimiter: ;

31: b[0] = 0.05;

31: ID, name= b

31: delimiter: [

31: NUM, val= 0

31: delimiter: ]

31: operator: =

31: NUM, val= 0.05

31: delimiter: ;

32: if(b[0] == 0.05)

32: reserved word: if

32: delimiter: (

32: ID, name= b

32: delimiter: [

32: NUM, val= 0

32: delimiter: ]

32: operator: ==

32: NUM, val= 0.05

32: delimiter: )

33: write(1);

33: reserved word: write

33: delimiter: (

33: NUM, val= 1

33: delimiter: )

33: delimiter: ;

34: else

34: reserved word: else

35: write(0);

35: reserved word: write

35: delimiter: (

35: NUM, val= 0

35: delimiter: )

35: delimiter: ;

36: }

36: delimiter: }

37: /\*output:

38: 1

39: 1

40: \*/

41: //:~

42: EOF

Process finished with exit code 0

4、

//**test2.cmm**

/\*\*  
 \* ���� һ�������ֵ  
 \* 2010-11-13  
 \* ������  
 \*   
 \* ���� һ�������ֵ ��Ҫ�ǣ�  
 \* 1.����ʱ��ֵ  
 \* 2.�������㸳ֵ  
 \* 3.read��ֵ  
 \* 4.�������ݣ�����һ��������ֵ������һ������  
 \* �����漰������ת������  
 \*/  
  
///////////////����ʱ��ֵ  
int aa = 23,ab = (-4);  
double b ,c =2.55;  
  
  
/////////////�������㸳ֵ  
int a = 15 - 3 \* ( 150 / 3 / 10);  
if(a == 0)  
{  
 write(1);  
}  
else  
 write(0);  
  
/////////////read��ֵ  
double r;  
read(r);  
write(r);  
  
//��������  
double r2 = 5;  
int a2 = 3;  
r2 = a2;  
if(r2 == 3)  
{  
 write(1);  
}  
else  
 write(0);  
  
r2 = a2 ;  
write(r2);  
/\*output:  
1  
233 #�������������  
233.0  
1  
3.0  
\*///:~

输出：

1: /\*\*

2: \* ���� һ�������ֵ

3: \* 2010-11-13

4: \* ������

5: \*

6: \* ���� һ�������ֵ ��Ҫ�ǣ�

7: \* 1.����ʱ��ֵ

8: \* 2.�������㸳ֵ

9: \* 3.read��ֵ

10: \* 4.�������ݣ�����һ��������ֵ������һ������

11: \* �����漰������ת������

12: \*/

13:

14: ///////////////����ʱ��ֵ

15: int aa = 23,ab = (-4);

15: reserved word: int

15: ID, name= aa

15: operator: =

15: NUM, val= 23

15: delimiter: ,

15: ID, name= ab

15: operator: =

15: delimiter: (

15: operator: -

15: NUM, val= 4

15: delimiter: )

15: delimiter: ;

16: double b ,c =2.55;

17:

18:

16: reserved word: double

16: ID, name= b

16: delimiter: ,

16: ID, name= c

16: operator: =

16: NUM, val= 2.55

16: delimiter: ;

19: /////////////�������㸳ֵ

20: int a = 15 - 3 \* ( 150 / 3 / 10);

20: reserved word: int

20: ID, name= a

20: operator: =

20: NUM, val= 15

20: operator: -

20: NUM, val= 3

20: operator: \*

20: delimiter: (

20: NUM, val= 150

20: operator: /

20: NUM, val= 3

20: operator: /

20: NUM, val= 10

20: delimiter: )

20: delimiter: ;

21: if(a == 0)

21: reserved word: if

21: delimiter: (

21: ID, name= a

21: operator: ==

21: NUM, val= 0

21: delimiter: )

22: {

22: delimiter: {

23: write(1);

23: reserved word: write

23: delimiter: (

23: NUM, val= 1

23: delimiter: )

23: delimiter: ;

24: }

24: delimiter: }

25: else

25: reserved word: else

26: write(0);

27:

26: reserved word: write

26: delimiter: (

26: NUM, val= 0

26: delimiter: )

26: delimiter: ;

28: /////////////read��ֵ

29: double r;

29: reserved word: double

29: ID, name= r

29: delimiter: ;

30: read(r);

30: reserved word: read

30: delimiter: (

30: ID, name= r

30: delimiter: )

30: delimiter: ;

31: write(r);

32:

31: reserved word: write

31: delimiter: (

31: ID, name= r

31: delimiter: )

31: delimiter: ;

33: //��������

34: double r2 = 5;

34: reserved word: double

34: ID, name= r2

34: operator: =

34: NUM, val= 5

34: delimiter: ;

35: int a2 = 3;

35: reserved word: int

35: ID, name= a2

35: operator: =

35: NUM, val= 3

35: delimiter: ;

36: r2 = a2;

36: ID, name= r2

36: operator: =

36: ID, name= a2

36: delimiter: ;

37: if(r2 == 3)

37: reserved word: if

37: delimiter: (

37: ID, name= r2

37: operator: ==

37: NUM, val= 3

37: delimiter: )

38: {

38: delimiter: {

39: write(1);

39: reserved word: write

39: delimiter: (

39: NUM, val= 1

39: delimiter: )

39: delimiter: ;

40: }

40: delimiter: }

41: else

41: reserved word: else

42: write(0);

43:

42: reserved word: write

42: delimiter: (

42: NUM, val= 0

42: delimiter: )

42: delimiter: ;

44: r2 = a2 ;

44: ID, name= r2

44: operator: =

44: ID, name= a2

44: delimiter: ;

45: write(r2);

45: reserved word: write

45: delimiter: (

45: ID, name= r2

45: delimiter: )

45: delimiter: ;

46: /\*output:

47: 1

48: 233 #�������������

49: 233.0

50: 1

51: 3.0

52: \*///:~

53: EOF

Process finished with exit code 0

5、

//**test7\_IF-ELSEWHILE.cmm**

/\*\*  
 \* IF-ELSE �� WHILE ���Ƕ�� ����  
 \* 2010-11-13  
 \* ������  
 \*   
 \* IF-ELSE �� WHILE ���Ƕ�� ���� ��Ҫ�ǣ�  
 \* 1.�����ж�  
 \* 2.���ִ��  
 \* 3.IF-ELSE �� WHILE ��� Ƕ��  
 \*/  
  
int a = 4;  
while(a <> 0)  
{  
 int j = a;  
 while(j <> 0)  
 {  
 if(j/2 <> 1)  
 write(j);  
 j = j-1;   
 }  
 if( a < 2)  
 {  
 write(a);  
 }  
 else  
 write(a+3);  
 a = a -1;  
}  
/\*output:  
4  
1  
7  
1  
6  
1  
5  
1  
1  
\*///:~

输出：

1: /\*\*

2: \* IF-ELSE �� WHILE ���Ƕ�� ����

3: \* 2010-11-13

4: \* ������

5: \*

6: \* IF-ELSE �� WHILE ���Ƕ�� ���� ��Ҫ�ǣ�

7: \* 1.�����ж�

8: \* 2.���ִ��

9: \* 3.IF-ELSE �� WHILE ��� Ƕ��

10: \*/

11:

12: int a = 4;

12: reserved word: int

12: ID, name= a

12: operator: =

12: NUM, val= 4

12: delimiter: ;

13: while(a <> 0)

13: reserved word: while

13: delimiter: (

13: ID, name= a

13: operator: <

13: operator: >

13: NUM, val= 0

13: delimiter: )

14: {

14: delimiter: {

15: int j = a;

15: reserved word: int

15: ID, name= j

15: operator: =

15: ID, name= a

15: delimiter: ;

16: while(j <> 0)

16: reserved word: while

16: delimiter: (

16: ID, name= j

16: operator: <

16: operator: >

16: NUM, val= 0

16: delimiter: )

17: {

17: delimiter: {

18: if(j/2 <> 1)

18: reserved word: if

18: delimiter: (

18: ID, name= j

18: operator: /

18: NUM, val= 2

18: operator: <

18: operator: >

18: NUM, val= 1

18: delimiter: )

19: write(j);

19: reserved word: write

19: delimiter: (

19: ID, name= j

19: delimiter: )

19: delimiter: ;

20: j = j-1;

20: ID, name= j

20: operator: =

20: ID, name= j

20: operator: -

20: NUM, val= 1

20: delimiter: ;

21: }

21: delimiter: }

22: if( a < 2)

22: reserved word: if

22: delimiter: (

22: ID, name= a

22: operator: <

22: NUM, val= 2

22: delimiter: )

23: {

23: delimiter: {

24: write(a);

24: reserved word: write

24: delimiter: (

24: ID, name= a

24: delimiter: )

24: delimiter: ;

25: }

25: delimiter: }

26: else

26: reserved word: else

27: write(a+3);

27: reserved word: write

27: delimiter: (

27: ID, name= a

27: operator: +

27: NUM, val= 3

27: delimiter: )

27: delimiter: ;

28: a = a -1;

28: ID, name= a

28: operator: =

28: ID, name= a

28: operator: -

28: NUM, val= 1

28: delimiter: ;

29: }

29: delimiter: }

30: /\*output:

31: 4

32: 1

33: 7

34: 1

35: 6

36: 1

37: 5

38: 1

39: 1

40: \*///:~

41: EOF

Process finished with exit code 0