结果正确

输出结果为：

\*\*\*\*\*\*\*\*\*\*\*start\*\*\*\*\*\*\*\*\*\*\*

Const: const

Int: int

customObj: x

assign: =

aNum: 0

semicolon: ;

Const: const

Char: char

customObj: y

assign: =

aChar: b

semicolon: ;

Int: int

customObj: z

semicolon: ;

Char: char

customObj: m

semicolon: ;

Int: int

customObj: foo

lParent: (

Int: int

customObj: a

comma: ,

Int: int

customObj: b

rParent: )

lBrace: {

If: if

lParent: (

customObj: a

Big: >

customObj: b

rParent: )

Printf: printf

lParent: (

aString: big

rParent: )

semicolon: ;

If: if

lParent: (

customObj: a

small: <

customObj: b

rParent: )

Printf: printf

lParent: (

aString: less

rParent: )

semicolon: ;

If: if

lParent: (

customObj: a

bigAndEql: >=

customObj: b

rParent: )

Printf: printf

lParent: (

aString: big or equal

rParent: )

semicolon: ;

If: if

lParent: (

customObj: a

smallAndEql: <=

customObj: b

rParent: )

Printf: printf

lParent: (

aString: less or equal

rParent: )

semicolon: ;

If: if

lParent: (

customObj: a

customObj: !

assign: =

customObj: b

rParent: )

Printf: printf

lParent: (

aString: not equal

rParent: )

semicolon: ;

If: if

lParent: (

customObj: a

eql: ==

customObj: b

rParent: )

Printf: printf

lParent: (

aString: equal

rParent: )

semicolon: ;

customObj: return

lParent: (

aNum: 233

rParent: )

semicolon: ;

rBrace: }

Void: void

customObj: foo2

lParent: (

rParent: )

lBrace: {

Int: int

customObj: i

semicolon: ;

For: for

lParent: (

customObj: i

assign: =

aNum: 0

semicolon: ;

customObj: i

small: <

aNum: 10

semicolon: ;

customObj: i

assign: =

customObj: i

plus: +

aNum: 1

rParent: )

lBrace: {

Printf: printf

lParent: (

customObj: i

rParent: )

semicolon: ;

rBrace: }

Do: do

lBrace: {

Printf: printf

lParent: (

customObj: i

rParent: )

semicolon: ;

customObj: i

assign: =

customObj: i

sub: -

aNum: 1

semicolon: ;

rBrace: }

While: while

lParent: (

customObj: i

Big: >

aNum: 0

rParent: )

semicolon: ;

rBrace: }

Int: int

customObj: foo3

lParent: (

Int: int

customObj: i

rParent: )

lBrace: {

If: if

lParent: (

customObj: i

eql: ==

aNum: 10

rParent: )

customObj: return

lParent: (

customObj: i

rParent: )

semicolon: ;

customObj: return

lParent: (

customObj: i

plus: +

customObj: foo3

lParent: (

customObj: i

plus: +

aNum: 1

rParent: )

rParent: )

semicolon: ;

rBrace: }

Void: void

customObj: foo4

lParent: (

Int: int

customObj: a

comma: ,

Int: int

customObj: b

rParent: )

lBrace: {

Printf: printf

lParent: (

customObj: a

sub: -

customObj: b

rParent: )

semicolon: ;

Printf: printf

lParent: (

customObj: a

plus: +

customObj: b

rParent: )

semicolon: ;

Printf: printf

lParent: (

customObj: a

mul: \*

customObj: b

rParent: )

semicolon: ;

Printf: printf

lParent: (

customObj: a

div: /

customObj: b

rParent: )

semicolon: ;

rBrace: }

Void: void

customObj: main

lParent: (

rParent: )

lBrace: {

Int: int

customObj: a

semicolon: ;

Char: char

customObj: b

semicolon: ;

Scanf: scanf

lParent: (

customObj: b

rParent: )

semicolon: ;

customObj: a

assign: =

customObj: foo

lParent: (

aNum: 2

comma: ,

aNum: 3

rParent: )

semicolon: ;

customObj: foo2

lParent: (

rParent: )

semicolon: ;

customObj: foo3

lParent: (

aNum: 1

rParent: )

semicolon: ;

customObj: foo4

lParent: (

aNum: 1

comma: ,

aNum: 2

rParent: )

semicolon: ;

customObj: return

semicolon: ;

rBrace: }

\*\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*\*\*