Tag

Token

+ tag : int

+ *toString()* *: string* <virtual>

Array

+ size = 1 : int

+ of : sh\_p<Type>

+ *toString()* *: string* <override>

Type

+ width = 0 : int

+ numeric(sh\_p<Type>) <static>

+ max(sh\_p<Type>, sh\_p<Type>) <static>

+ Int : sh\_p<Type> <static>

+ Float : sh\_p<Type> <static>

+ Char : sh\_p<Type> <static>

+ Bool : sh\_p<Type> <static>

Word

+ lexeme : string

+ toString() : string <override>

+ and : shared\_ptr<Word> <static>

+ or : shared\_ptr<Word> <static>

+ eq : shared\_ptr<Word> <static>

+ ne : shared\_ptr<Word> <static>

+ le : shared\_ptr<Word> <static>

+ ge : shared\_ptr<Word> <static>

+ minus : shared\_ptr<Word> <static>

+ True : shared\_ptr<Word> <static>

+ False : shared\_ptr<Word> <static>

+ temp : shared\_ptr<Word> <static>

Real

+ value : float

+ *toString()* *: string* <override>

Num

+ value : int

+ *toString() : string* <override>

Node

Expr

Stmt

Test

Env

Lexer

Parser

Tag

Else

Set

Seq

SetElem

Break

Continue

And

Rel

Or

Not

If

For

While

Arith

Do

Array

Type

Token

Num

Word

Real

Access

Unary

Constant

Logical

Temp

Op

Id

Not

+ *jumping(int, int) : void* <override>

+ *toString() : string* <override>

And

+ *jumping(int, int) : void* <override>

Rel

+ *check(sh\_p<Type>, sh\_p<Type> : sh\_p<Type>* <override>

+ *jumping(int, int) : void* <override>

Or

+ *jumping(int, int) : void* <override>

Arith

- expr1 : sh\_p<Expr>

- expr2 : sh\_p<Expr>

+ *gen() : sh\_p<Expr>* <override>

+ *toString()* *: string* <override>

Logical

+ expr1 : sh\_p<Expr>

+ expr2 : sh\_p<Expr>

+ *check(sh\_p<Type>, sh\_p<Type>)* *: sh\_p<Type>* <virtual>

+ *gen() : sh\_p<expr>* <override>

+ *toString()* *: string*  <override>

Constant

+ *jumping(int, int)* *: void* <override>

+ TRUE : sh\_p<Constant> <static>

+ FALSE : sh\_p<Constant> <static>

Temp

+ count = 0 : int <static>

+ number = 0 : int

+ *toString()* *: string* <override>

Op

+ *reduce()* *: sh\_p<Expr>* <override>

Access

+ arr : sh\_p<Id>

+ index : sh\_p<Expr>

+ *gen() : sh\_p<Expr>* <override>

+ *jumping(int, int)* *: void* <override>

+ *toString()* *: string* <override>

Unary

- expr : sh\_p<Expr>

+ *gen()* *: sh\_p<Expr>* <override>

+ *toString()* *: string*  <override>  
+

Id

+ offset : int

Expr

+ op : sh\_p<Token>

+ type : sh\_p<Type>

+ *gen() : sh\_p<Expr>* <virtual>

+ *reduce() : sh\_p<Expr>* <virtual>

+ *jumping(int, int)* *: void* <virtual>

+ emitJumps(string, int, int) : void

+ getptr() : sh\_p<Expr>

+ *toString() : string*  <virtual>

Node

+ lexline : int <static>

+ labels : int <static>

+ newLabel() : int

+ emitLabel(int) : void

+ emit(string) : void

+ error(string) : void

Node

Break

- stmt : sh\_p<Stmt>

+ *gen(int, int) : void*  <override>

Continue

- stmt : sh\_p<Stmt>

+ *gen(int, int) : void*  <override>

Set

+ id : sh\_p<Id>

+ expr : sh\_p<Expr>

+ check(sh\_p<Type>, sh\_p<Type> : sh\_p<Type>

+ *gen(int, int) : void* <override>

Seq

- stmt1 : sh\_p<Stmt>

- stmt2 : sh\_p<Stmt>

+ *gen(int, int) : void*  <override>

If

- expr : sh\_p<Expr>

- stmt : sh\_p<Stmt>

+ *gen(int , int) : void* <override>

Else

- expr : sh\_p<Expr>

- stmt1 : sh\_p<Stmt>

- stmt2 : sh\_p<Stmt>

+ *gen(int, int) : void* <override>

SetElem

- arr : sh\_p<Id>

- index : sh\_p<Expr>

- expr : sh\_p<Expr>

+ check(sh\_p<Type>, sh\_p<Type> : sh\_p<type>

+ *gen(int, int) : void*  <override>

While

- stmt : sh\_p<Stmt>

- expr : sh\_p<Expr>

+ init(sh\_p<Stmt>, sh\_p<Expr> : void

+ *gen(int, int) : void*  <override>

For

- init\_val : sh\_p<Stmt>

- condition : sh\_p<Expr>

- step : sh\_p<Stmt>

- instruction : sh\_p<Stmt>

- variable : sh\_p<Id>

+ init(sh\_p<Stmt>, sh\_p<Expr>, sh\_p<Stmt>, sh\_p<Stmt> : void

+ *gen(int, int) : void*  <override>

Do

- stmt : sh\_p<Stmt>

- expr : sh\_p<Expr>

+ init(sh\_p<Stmt>, sh\_p<Expr> : void

+ *gen(int, int) : void*  <override>

Stmt

+ begin = 0 : int

+ after = 0 : int

+ Null : sh\_p<Stmt> <static>

+ Enclosing = Null : sh\_p<Stmt> <static>

+ *gen() : void* <virtual>