Contents

1

class **MMCls**

Memory Management

Public Members

Memory Management. Keeps track of everything that is necessary to assign memory to data items.

1.1

int **allocate** (int size)

Allocates memory of size memory locations

Allocates memory of size memory locations.

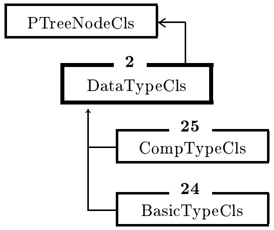
Return Value: Base address
Parameters: `size` Number of memory locations

2

class **DataTypeCls** : public PTreeNodeCls

Abstract base class for all data types

Inheritance



Public Members

2.1	int	**getSize** ()	*Returns the memory size of this data type*	3
2.2	PDataTypeCls	**copy** ()	*Returns a deep copy of this data type*	4

Abstract base class for all data types.

2.1

int **getSize** ()

Returns the memory size of this data type

Returns the memory size of this data type.

Parameters: `Amout` of memory units

2.2

PDataTypeCls **copy** ()

Returns a deep copy of this data type

Returns a deep copy of this data type.

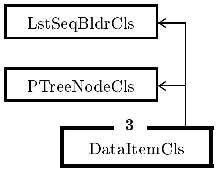
Return Value: Copy of data type

3

class **DataItemCls** : public PTreeNodeCls, public LstSeqBldrCls

A data item

Inheritance



Public Members

A data item. Like integer, array slice, etc.

```
DataItem: <Identifier>
        | <DataItem> [ Expr ]
```


3.1

DataItemCls (PPTreeNodeCls item, PPTreeNodeCls sub=NULL)

Constructor

Constructor.

Parameters:

`item`	Data item
`sub`	Data qualifier; may be *NULL* for basic types (default).

3.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

Basic version. Emits the base address and the modifier, if any. Can only handle basic types and arrays of basic types.

Return Value: 0

3.3

PDataTypeCls **typeEmit** ()

SLICK Emitter

SLICK Emitter.

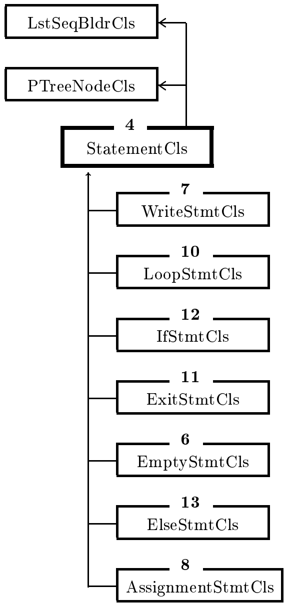
Full version. Emits the base address and recursively applies all modifiers to get real memory address onto stack.

4

class **StatementCls** : public PTreeNodeCls, public LstSeqBldrCls

The base class for all statements

Inheritance



Public Members

4.1 **StatementCls** () *Constructor* 9

4.2 **StatementCls** (char *StmtText)

Protected Members

The base class for all statements.

See Also: EmptyStmtCls
AssignmentStmt
WriteStmt
LoopStmtCls
ExitStmtCls
IfStmtCls (→12, *page 25*)

4.1

StatementCls ()

Constructor

Constructor.

4.2

StatementCls (char *StmtText)

Constructor

Constructor.

Parameters: `StmtText` Input text for this statement

4.3

int **emit** ()

SLICK Emitter

SLICK Emitter.

Nothing is emitted.

Return Value: 0

4.4

char* **stmt_text**

The input text of this statement

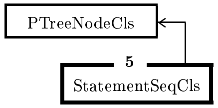
The input text of this statement

5

class **StatementSeqCls** : public PTreeNodeCls

A sequence of statements

Inheritance



Public Members

A sequence of statements.

```
StatementSeq: <Statement>
            | <StatementSeq> ; <Statement>
```

5.1

StatementSeqCls (PPTreeNodeCls Stmt)

Constructor

Constructor.

Parameters: `Stmt` First statement in this new list

5.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

Emits all statements in the list.

Return Value: 0

5.3

PPTreeNodeCls **append** (PPTreeNodeCls Stmt)

Append another statement

Append another statement.

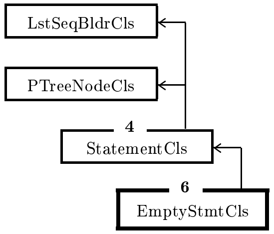
Parameters: `Stmt` Statement to append.

6

class **EmptyStmtCls** : public StatementCls

An empty statement

Inheritance



Public Members

6.1		**EmptyStmtCls** ()	*Constructor*	14
6.2	int	**emit** ()	*SLICK Emitter*	14

An empty statement. Ie, statement with no code.

`Statement:`

6.1

EmptyStmtCls ()

Constructor

Constructor.

6.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

Generates no code.

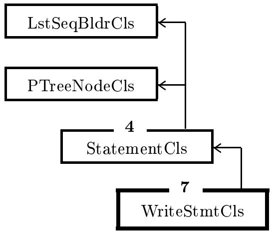
return 0

7

class **WriteStmtCls** : public StatementCls

A write statement

Inheritance



Public Members

A write statement.

```
WriteStmt: write ( <Expr> )
```

7.1

WriteStmtCls (PPTreeNodeCls Expr, char* StmtText)

Constructor

Constructor.

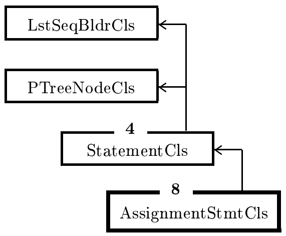
Parameters:

`Expr`	Expression to write
`StmtText`	Text of input file that created this statement

8 class **AssignmentStmtCls** : public StatementCls

An assignment statement

Inheritance



Public Members

An assignment statement.

```
AssignmentStmt: <AssignmentSeq> <Expr>
```


8.1 **AssignmentStmtCls** (PPTreeNodeCls assignSeq, PPTreeNodeCls Expr, char* StmtText)

Constructor

Constructor.

Parameters:

`assignSeq`	Data items to assign expression to
`Expr`	The expression
`StmtText`	Text of input file that created this assignment

8.2 int **emit** ()

SLICK Emitter

SLICK Emitter.

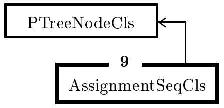
Emits the data items and the expression. Then, for each data item an assignment opcode is emitted. Finally, the expression is *pop*ed off the stack.

Return Value: 0

9 class **AssignmentSeqCls** : public PTreeNodeCls

The left side of multiple assignments

Inheritance



Public Members

The left side of multiple assignments.

This is the sequence of variables the expression to the right should be assigned to.

```
AssignmentSeq: <AssignmentSeq> <DataItem> :=
             | <DataItem> :=
```


9.1 **AssignmentSeqCls** (PPTreeNodeCls item)

Constructor

Constructor.

Parameters: `item` First data item (ie, lvalue) to assign value to

9.2 int **emit** ()

SLICK Emitter

SLICK Emitter.

Emits all data items (ie, lvalues)

Return Value: 0

9.3 PPTreeNodeCls **append** (PPTreeNodeCls item)

Append another data item to the sequence

Append another data item to the sequence.

Return Value: Pointer to sequence, including current sequence and the added item

Parameters: `item` Data item (ie, lvalue) to append

9.4

int **getLength** ()

Get the number of data items in sequence

Get the number of data items in sequence.

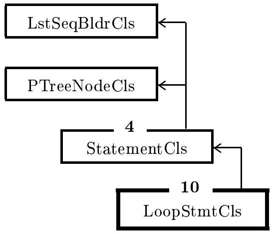
Return Value: Number of data items in sequence.

10

class **LoopStmtCls** : public StatementCls

Loop statement

Inheritance



Public Members

Loop statement.

```
Statement: loop <StatementSeq> end loop
```

10.1

LoopStmtCls (PPTreeNodeCls stmts)

Constructor

Constructor.

Parameters: `stmts` Statements to loop

10.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

Emits *loop* opcode, the statements and *endloop* opcode.

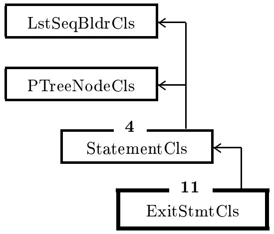
Return Value: 0

11

class **ExitStmtCls** : public StatementCls

Exit statement

Inheritance



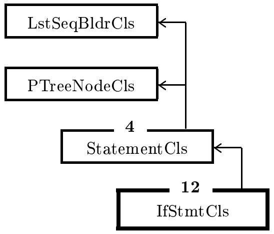
Exit statement.

`Statement: exit`

12 class **IfStmtCls** : public StatementCls

If statement

Inheritance



Public Members

If statement.

```
Statement: if <Condition> then <StatementSeq> <ElseBlock> end if
```

See Also: ElseStmtCls (→13, *page 27*)

12.1 **IfStmtCls** (PPTreeNodeCls condition, PPTreeNodeCls statements, PPTreeNodeCls elseStmts)

Constructor

Constructor.

Parameters:

`condition`	Condition for if statement
`statements`	Statements for true condition
`elseStmts`	Statements for false condition

12.2 int **emit** ()

SLICK Emitter

SLICK Emitter.

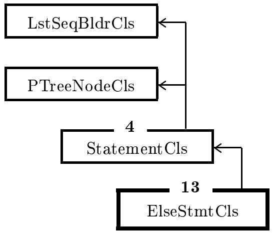
Emit the condition and *test* it. Then ouput the statements and the else block and close with *endif*.

Return Value: 0

13 class **ElseStmtCls** : public StatementCls

Else block

Inheritance



Public Members

Else block.

```
ElseBlock:
| else <StatementSeq>
```

See Also: IfStmtCls (→12, *page 25*)

13.1 **ElseStmtCls** ()

Constructor

Constructor.

Creates an empty else block.

13.2 **ElseStmtCls** (PPTreeNodeCls statements)

Constructor

Constructor.

Parameters: `statements` Statements in the block.

13.3 int **emit** ()

SLICK Emitter

SLICK Emitter.

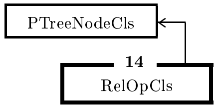
Emits nothing for an empty else, or the *else* opcode and the statements.

Return Value: 0

14 class **RelOpCls** : public PTreeNodeCls

A relational operator

Inheritance



Public Members

A relational operator.

```
RelOp: <
       | <=
       | =
       | <>
       | >
       | >=
```


14.1 **RelOp**

The relational operators

The relational operators.

ltOp less than

leOp less or equal

eqOp equal

neOp not equal

geOp greater or equal

gtOp greater than

14.2 **RelOpCls** (RelOp op)

Constructor

Constructor.

Parameters: `op` Relational operator

14.3

int **emit** ()

SLICK Emitter

SLICK Emitter.

Emits the SLICK opcode of the operator.

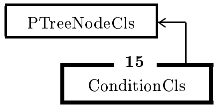
Return Value: 0

15

class **ConditionCls** : public PTreeNodeCls

A condition

Inheritance



Public Members

A condition.

```
Condition: <Expr> <RelOp> <Expr>
```

15.1

ConditionCls (PPTreeNodeCls expr1, PPTreeNodeCls relOp, PPTreeNodeCls expr2)

Constructor

Constructor.

Parameters:

`expr1`	Left expression
`relOp`	Relational operator
`expr2`	Right expression

15.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

First, both expressions are emitted, then the relational operator.

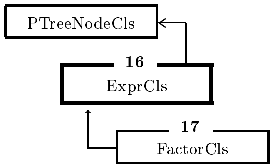
Return Value: 0

16

class **ExprCls** : public PTreeNodeCls

Base class for an expression

Inheritance



Public Members

Base class for an expression.

See Also: FactorCls (→17, *page 36*)

16.1

ExprCls ()

Constructor

Constructor.

16.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

Nothing emitted.

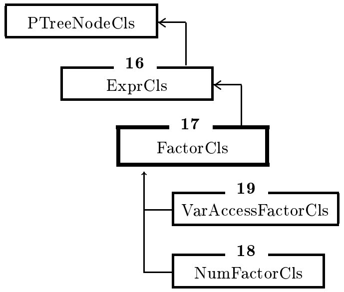
Return Value: 0

17

class **FactorCls** : public ExprCls

The base class for factors

Inheritance



Public Members

The base class for factors.

See Also: NumFactorCls
VarAccessFactorCls (→19, *page 40*)

17.1

FactorCls ()

Constructor

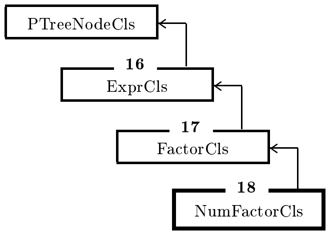
Constructor.

18

class **NumFactorCls** : public FactorCls

A number, used as a factor in an expression

Inheritance



Public Members

A number, used as a factor in an expression.

```
Factor: <Number>
```

18.1

NumFactorCls (PPTreeNodeCls NumLit)

Constructor

Constructor.

Parameters: `NumLit` Numeric literal of the number.

18.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

*push*es the literal onto the stack.

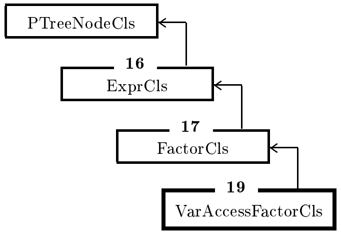
Return Value: 0

19

class **VarAccessFactorCls** : public FactorCls

The value of a data item

Inheritance



Public Members

The value of a data item.

```
Factor: <DataItem>
```

19.1

VarAccessFactorCls (PPTreeNodeCls Item)

Constructor

Constructor.

Parameters: `Item` Data item

19.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

Emits the data item and the *getValue* opcode.

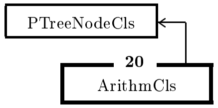
Return Value: 0

20

class **ArithmCls** : public PTreeNodeCls

An arithmetic expression

Inheritance



Public Members

An arithmetic expression.

```
Expr: Expr + Expr
    | Expr - Expr
    | Expr * Expr
    | Expr / Expr
```

20.1

ArithmOp

The available arithmetic operations

The available arithmetic operations.

PlusOp Addition

MinusOp Subtraction

MultOp Multiplication

DivOp Division

20.2

ArithmCls (PPTreeNodeCls left, PPTreeNodeCls right, ArithmOp op)

Constructor

Constructor.

Parameters:	`left`	Left side of expression
	`right`	Right side of expression
	`op`	Arithmetic operation

20.3

int **emit** ()

SLICK Emitter

SLICK Emitter.

Emits the left, then the right expression and finally the operator opcode (polish notation).

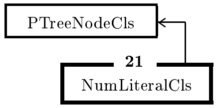
Return Value: 0

21

class **NumLiteralCls** : public PTreeNodeCls

A numeric literal

Inheritance



Public Members

A numeric literal.

```
Number: {digits}
```

21.1

NumLiteralCls ()

Constructor

Constructor.

The literal is taken from LexToken.

See Also: LexTokCls

21.2

int **get_value** ()

Returns the value of the literal

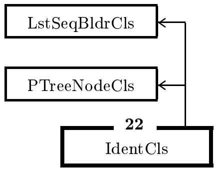
Returns the value of the literal.

Return Value: The value of the literal

22 class **IdentCls** : public PTreeNodeCls, public LstSeqBldrCls

An Identifier

Inheritance



Public Members

An Identifier.

```
Identifier: {letter}{letter_or_digit}*
```

22.1 **IdentCls** ()

Constructor

Constructor.

The name of the identifier is stored in the LexToken.

See Also: PTreeNodeCls

22.2 char* **get_name** ()

Returns the name of the identifier

Returns the name of the identifier.

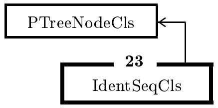
Return Value: The name of the identifier.

23

class **IdentSeqCls** : public PTreeNodeCls

A sequence of identifiers

Inheritance



Public Members

23.1		**IdentSeqCls** (PPTreeNodeCls ident)	*Constructor*	50
23.2	int	**emit** ()	*SLICK Emitter*	50
23.3	PPTreeNodeCls	**append** (PPTreeNodeCls ident)	*Appends another identifier*	50
23.4	PIdentCls	**get_first** ()	*Returns the first identifier in the sequence*	51

A sequence of identifiers.

```
IdentifierSeq: <IdentifierSeq> , <Identifier>
             | <Identifier>
```

23.1

IdentSeqCls (PPTreeNodeCls ident)

Constructor

Constructor.

Parameters: `ident` First identifier of the new sequence.

23.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

Nothing emitted.

Return Value: 0

23.3

PPTreeNodeCls **append** (PPTreeNodeCls ident)

Appends another identifier

Appends another identifier.

Parameters: `ident` Identifer to append

23.4

PIdentCls **get_first** ()

Returns the first identifier in the sequence

Returns the first identifier in the sequence.

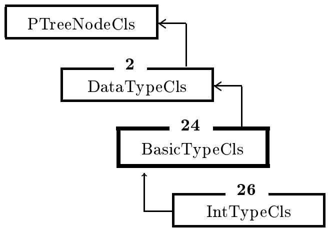
Parameters: `First` identifier in the sequence.

24

class **BasicTypeCls** : public DataTypeCls

Base class for all basic types

Inheritance



Base class for all basic types.

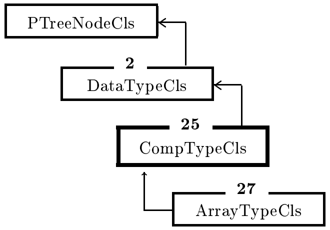
See Also: IntTypeCls (→26, *page 54*)

25

class **CompTypeCls** : public DataTypeCls

Base class for all compound types

Inheritance



Base class for all compound types.

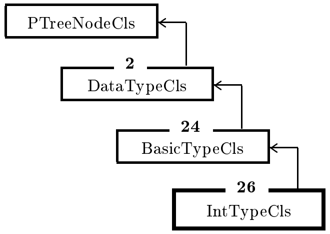
See Also: ArrayTypeCls (→27, *page 56*)

26

class **IntTypeCls** : public BasicTypeCls

The integer type

Inheritance



Public Members

26.1	int	**getSize** ()	*The memory size of a single integer is 1*	55
26.2	PIntTypeCls	**copy** ()	*Returns a deep copy of this data type*	55

The integer type.

```
BasicType: {digits}
```

26.1

int **getSize** ()

The memory size of a single integer is 1

The memory size of a single integer is 1.

Return Value: 1

26.2

PIntTypeCls **copy** ()

Returns a deep copy of this data type

Returns a deep copy of this data type.

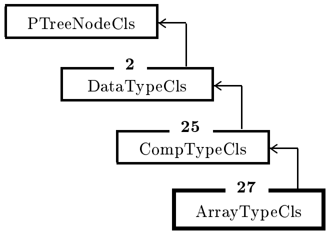
Return Value: Copy of data type

27

class **ArrayTypeCls** : public CompTypeCls

The array type

Inheritance



Public Members

The array type.

```
CompoundType: array [ <Number> .. <Number> ] of <DataType>
```

27.1

ArrayTypeCls (PPTreeNodeCls rangeFrom, PPTreeNodeCls rangeTo, PPTreeNodeCls dataType)

Constructor

Constructor.

Parameters:

`rangeFrom`	First index of array
`rangeTo`	Last index of array
`dataType`	Data type of array elements

27.2

int **getSize** ()

Returns the memory size of this array type

Returns the memory size of this array type.

Return Value: Memory size used by this type

27.3

int **getItemSize** ()

Returns the memory size of a single item

Returns the memory size of a single item.

Return Value: Element memory size

27.4

PNumLiteralCls **getRangeFrom** ()

Returns the first index

Returns the first index.

Return Value: First index

27.5

PNumLiteralCls **getRangeTo** ()

Returns the last index

Returns the last index.

Return Value: Last index

27.6

PDataTypeCls **getItemDataType** ()

Returns the element data type

Returns the element data type.

Return Value: Data type of elements

27.7

PArrayTypeCls **copy** ()

Returns a deep copy of this data type

Returns a deep copy of this data type.

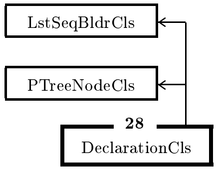
Return Value: Copy of data type

28

class **DeclarationCls** : public PTreeNodeCls, public LstSeqBldrCls

A declaration

Inheritance



Public Members

A declaration.

```
Declaration: <IdentifierSeq> : <DataType>
                   |
```

28.1

DeclarationCls ()

Constructor

Constructor.

Creates an empty declartion.

28.2

DeclarationCls (PPTreeNodeCls identSeq, PPTreeNodeCls dataType, char *declText)

Constructor

Constructor.

Parameters:

`identSeq`	Sequence of identifiers
`dataType`	Array element data type
`declText`	Input text that created this declaration

28.3

int **emit** ()

SLICK Emitter

SLICK Emitter.

Nothing emitted.

Return Value: 0

28.4

PIdentSeqCls **ident_seq**

Sequence of identifiers

Sequence of identifiers.

28.5

PDataTypeCls **data_type**

Element data type

Element data type.

28.6

char* **decl_text**

declaration text from input

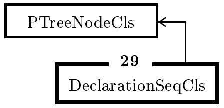
declaration text from input

29

class **DeclarationSeqCls** : public PTreeNodeCls

Sequence of declarations

Inheritance



Public Members

Sequence of declarations.

```
DeclarationSeq: <DeclarationSeq> ; <Declaration>
              | <Declaration>
```

29.1

DeclarationSeqCls ()

Constructor

Constructor.

Empty, ie no, declarations.

29.2

DeclarationSeqCls (PPTreeNodeCls decl)

Constructor

Constructor.

Parameters: `decl` First declaration for this new sequence.

29.3

int **emit** ()

SLICK Emitter

SLICK Emitter.

Nothing emitted.

Return Value: 0

29.4

PPTreeNodeCls **append** (PPTreeNodeCls decl)

Appends another declaration

Appends another declaration.

Parameters: `decl` Declaration to append

30

class **AddressCls**

The memory address of an data item

Public Members

Protected Members

The memory address of an data item.

See Also: MMCls (→1, *page 2*)

30.1

AddressCls ()

Constructor

Constructor.

30.2

void **setAddr** (int a)

Sets the address

Sets the address.

Parameters: `a` Address

30.3

int **getAddr** ()

Returns the address

Returns the address.

Return Value: Address

30.4

int **addr**

Memory address

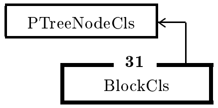
Memory address.

31

class **BlockCls** : public PTreeNodeCls

The program statements block

Inheritance



Public Members

The program statements block.

```
Block: begin <StatementSeq> end .
```

31.1

BlockCls (PPTreeNodeCls StmtSeq)

Constructor

Constructor.

Parameters: `StmtSeq` Sequence of statements for the program.

31.2

int **emit** ()

SLICK Emitter

SLICK Emitter.

Emits the program sequence.

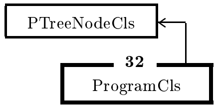
Return Value: Status of program sequence emit.

32

class **ProgramCls** : public PTreeNodeCls

A epc2 program

Inheritance



Public Members

A epc2 program.

```
Program: program <Identifier> ; <DeclarationBlock> <Block>
```

32.1

ProgramCls ()

Constructor

Constructor.

32.2

ProgramCls (PPTreeNodeCls Ident, PPTreeNodeCls Block)

Constructor

Constructor.

Parameters: `Ident` Program name
`Block` Codeblock

32.3

int **emit** ()

SLICK Emitter

SLICK Emitter.

Emits the code and *finish* opcode.

Return Value: 0

32.4

void **print** ()

Does nothing

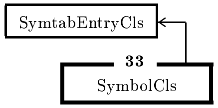
Does nothing.

33

class **SymbolCls** : public SymtabEntryCls

A symbol

Inheritance



Public Members

A symbol. Includes identifier, data type and memory address.

33.1

SymbolCls ()

Constructor

Constructor.

33.2

SymbolCls (char *name, PDataTypeCls dataType, PAddressCls addr)

Constructor

Constructor.

Parameters:

	`name`	Symbol name
	`dataType`	Data type
	`addr`	Memory address

33.3

PDataTypeCls **getDataType** ()

Returns the data type

Returns the data type.

Return Value: Data type

33.4

PAddressCls **getAddress** ()

Returns the address

Returns the address.

Return Value: Address

Class Graph