Matrix Calculator

Generated by Doxygen 1.8.8

Sun Jan 4 2015 05:54:39

Contents

Chapter 1

Project documentation

1.1 Introduction

This is the documentation of our Matrix Calculator.

This project has been made by Axel Mousset and Aurélien Labate when they were following the NF05 course (Introduction to C) at the University of Technology of Troyes.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

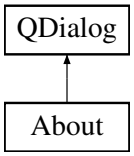
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 About Class Reference

Inheritance diagram for About:



Public Member Functions

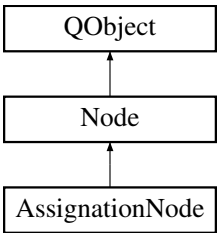
- **About** (QWidget ∗parent=0)

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/about.h
- /home/bob/dev/utt-nf05-project/sources/about.cpp

4.2 AssignationNode Class Reference

Inheritance diagram for AssignationNode:



Public Member Functions

- AssignationNode (VarNode ∗variable, ExpressionNode ∗expression)

  constructor

- ~AssignationNode ()
  destructor
- virtual Calculable * execute ()
  put expression value in memory as varName
- VarNode * getVariable () const
  variable Node accessor
- ExpressionNode * getExpression () const
  expression Node accessor
- virtual QString toString () const
  toString method

Protected Attributes

- VarNode * **variable**
- ExpressionNode * **expression**

4.2.1 Constructor & Destructor Documentation

4.2.1.1 AssignationNode::AssignationNode (VarNode * *variable*, ExpressionNode * *expression*)

constructor

Parameters

variable	VarNode to set
expression	expression to execute

4.2.2 Member Function Documentation

4.2.2.1 Calculable * AssignationNode::execute ( ) [virtual]

put expression value in memory as varName

Returns

expression value

Implements Node.

4.2.2.2 ExpressionNode * AssignationNode::getExpression ( ) const

expression Node accessor

Returns

a pointer to the expression Node associated to this association Node

4.2.2.3 VarNode * AssignationNode::getVariable ( ) const

variable Node accessor

Returns

a pointer to the variable Node associated to this assocation Node

4.2.2.4 QString AssignationNode::toString ( ) const [virtual]

toString method

Returns

a QString representation of the Node

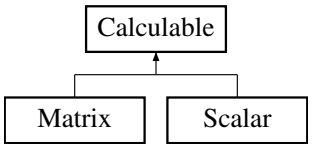
Implements Node.

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/assignationNode.h
- /home/bob/dev/utt-nf05-project/sources/lib/assignationNode.cpp

4.3 Calculable Class Reference

Inheritance diagram for Calculable:



Public Member Functions

- Calculable (QString value)
  constructor
- Calculable ()
  constructor
- ~Calculable ()
  destructor
- virtual QString getValue ()
  value accessor
- virtual void setValue (QString newValue)
  value setter
- virtual QString toString ()
  toString method
- virtual Calculable * operator* (Calculable &a)
  *overload operator * between two Calculable*
- virtual Calculable * operator/ (Calculable &a)
  overload operator / between two Calculable
- virtual Calculable * operator+ (Calculable &a)
  overload operator + between two Calculable
- virtual Calculable * operator- (Calculable &a)
  overload operator - between two Calculable
- virtual Calculable * operator% (Calculable &a)
  overload operator % between two Calculable
- virtual Calculable * operator^ (Calculable &a)
  overload operator ^ between two Calculable
- virtual std::string getTypeStr ()=0

Define the type of the element as a string.

- virtual TokenKind getType ()=0

  Define the type of the element as a TokenKind from token.h.

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Calculable::Calculable (QString *value*)

constructor

Parameters

value	the calculable value

4.3.1.2 Calculable::~Calculable ( )

destructor

[long description]

4.3.2 Member Function Documentation

4.3.2.1 QString Calculable::getValue ( ) [virtual]

value accessor

Returns

the value of the Calculable

Reimplemented in Matrix, and Scalar.

4.3.2.2 Calculable * Calculable::operator% (Calculable & *a*) [virtual]

overload operator % between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented in Scalar.

4.3.2.3 Calculable * Calculable::operator* (Calculable & *a*) [virtual]

overload operator * between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented in Matrix, and Scalar.

4.3.2.4 Calculable * Calculable::operator+ (Calculable & *a*) [virtual]

overload operator + between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented in Matrix, and Scalar.

4.3.2.5 Calculable * Calculable::operator- (Calculable & *a*) [virtual]

overload operator - between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented in Matrix, and Scalar.

4.3.2.6 Calculable * Calculable::operator/ (Calculable & *a*) [virtual]

overload operator / between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented in Scalar.

4.3.2.7 Calculable * Calculable::operator^ (Calculable & *a*) [virtual]

overload operator ^ between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented in Scalar.

4.3.2.8 void Calculable::setValue (QString *newValue*) [virtual]

value setter

Parameters

newValue	the value to set

Reimplemented in Matrix, and Scalar.

4.3.2.9 QString Calculable::toString ( ) [virtual]

toString method

Returns

a QString representation of the Node

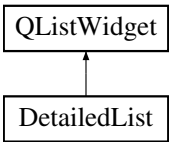
The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/calculable.h
- /home/bob/dev/utt-nf05-project/sources/lib/calculable.cpp

4.4 DetailedList Class Reference

A list widget with details when you click on the title line.

Inheritance diagram for DetailedList:



Public Slots

- void deleteAll ()

  Delete all elements from the list.

Signals

- void elementDeleted (QString title)

  Signal emitted when the user delete an element.

Public Member Functions

- DetailedList (QWidget *parent=0)

  Constructor.
- void addElement (QString title, QString content, bool expanded=true)

  Add a new element to the list.
- void deleteElement (QString title)

  Delete an element from the list.
- void UpdateElement (QString title, QString content)

  Update an element from the list.
- void expandElement (QString title, bool expand=true)

  Expand or reduce an element of the list.
- QMultiMap< QString, QListWidgetItem * > * **getList** ()

Protected Slots

- void itemClicked (QListWidgetItem *clicked)

  Slot triggered when a click is made on an item.
- void customContextMenuRequested (const QPoint &pos)

  Slot triggered when a right click is made.
- void deleteTriggered ()

  Slot triggered when a click is made on the delete button of the context menu.

4.4.1 Detailed Description

A list widget with details when you click on the title line.

A list widget with details when you click on the title line with a triangle or an arrow that indicate current state. Each element is composed by to QListWidgetItem : title and content. The list is sorted by title alphabetical order

Warning : As each element is indexed by the title, it has to be unique.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 DetailedList::DetailedList (QWidget * *parent* = `0`) `[explicit]`

Constructor.

Construct an empty DetailedList with the given *parent*

Parameters

parent	

4.4.3 Member Function Documentation

4.4.3.1 void DetailedList::addElement (QString *title*, QString *content*, bool *expanded* = `true`)

Add a new element to the list.

Add a new element to the list

Parameters

title	Plain text title that is allways visible
content	Rich text content that will be visible when the user click on the line
expanded	Choose if the new element will be expanded or not

4.4.3.2 void DetailedList::customContextMenuRequested (const QPoint & *pos*) [protected], [slot]

Slot triggered when a right click is made.

Slot triggered when a right click is made. This will add a context menu on elements that will allow user to delete the element.

Parameters

pos	Position of the mouse when the user right click

4.4.3.3 void DetailedList::deleteElement (QString *title*)

Delete an element from the list.

Delete an element from the list

Parameters

title	Plain text title of the element that you want to delete

4.4.3.4 void DetailedList::deleteTriggered ( ) [protected], [slot]

Slot triggered when a click is made on the delete button of the context menu.

Slot triggered when a click is made on the delete button of the context menu. This will emit the but public elementDeleted() signal and then remove the element from the list.

4.4.3.5 void DetailedList::elementDeleted (QString *title*) [signal]

Signal emitted when the user delete an element.

Signal emitted when the user delete an element

Parameters

title	The title of the deleted element

4.4.3.6 void DetailedList::expandElement (QString *title*, bool *expand* = true)

Expand or reduce an element of the list.

Expand or reduce an element of the list

Parameters

title	Plain text title of the element that you want to update
expand	Choose if the element will be expanded or not

4.4.3.7 void DetailedList::itemClicked (QListWidgetItem * *clicked*) [protected], [slot]

Slot triggered when a click is made on an item.

Slot triggered when a click is made on an item. This is used to expand and reduce elements when you click on a title.

Parameters

clicked	A pointer to the clicked item

4.4.3.8 void DetailedList::UpdateElement (QString *title*, QString *content*)

Update an element from the list.

Update an element from the list

Parameters

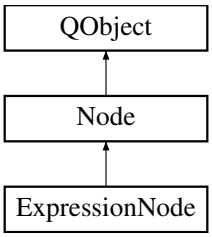
title	Plain text title of the element that you want to update
content	The new content of the element

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/detailedlist.hpp
- /home/bob/dev/utt-nf05-project/sources/detailedlist.cpp

4.5 ExpressionNode Class Reference

Inheritance diagram for ExpressionNode:



Public Member Functions

- ExpressionNode (QList< Token > expression)
  constructor
- ~ExpressionNode ()
  destructor
- Calculable * execute ()
  Execute the node.
- QString toString () const
  toString method

Protected Member Functions

- void convertToRPN ()
  Convert current expression to a RPN notation.

- bool isFunction (Token token)

  Determine if a token is a function or a variable.

Protected Attributes

- QList< Token > **expression**
- Calculable * **value**

4.5.1 Constructor & Destructor Documentation

4.5.1.1 ExpressionNode::ExpressionNode (QList< Token > *expression*)

constructor

Parameters

expression	the expression in tokens

4.5.2 Member Function Documentation

4.5.2.1 Calculable * ExpressionNode::execute ( ) [virtual]

Execute the node.

Returns

a Calculable pointer

Implements Node.

4.5.2.2 bool ExpressionNode::isFunction (Token *token*) [protected]

Determine if a token is a function or a variable.

Parameters

token	token

Returns

nature of the token

4.5.2.3 QString ExpressionNode::toString ( ) const [virtual]

toString method

Returns

a QString representation of the Node

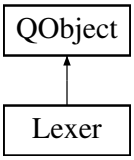
Implements Node.

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/expressionNode.h
- /home/bob/dev/utt-nf05-project/sources/lib/expressionNode.cpp

4.6 Lexer Class Reference

Inheritance diagram for Lexer:



Signals

- void **lexerError** (QString line, int offset)

Public Member Functions

- Lexer (QString source)
  constructor
- ~Lexer ()
  destructor
- QList< Token > run ()
  tokenize the source

Protected Member Functions

- Token match (QString line, int offset)
  try to find a token on a string, begenning on given offset

Static Protected Member Functions

- static QMap< TokenKind, QRegExp > initializeTokens ()
  Initialize Lexer::tokens static map.

Protected Attributes

- QString **source**

Static Protected Attributes

- static QMap< TokenKind, QRegExp > **tokens** = Lexer::initializeTokens()

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Lexer::Lexer (QString *source*)

constructor

Parameters

source	String going to be tokenized

4.6.2 Member Function Documentation

4.6.2.1 QMap< TokenKind, QRegExp > Lexer::initializeTokens ( ) [static], [protected]

Initialize Lexer::tokens static map.

Returns

Map of TokenKind and it's associated regex

4.6.2.2 Token Lexer::match (QString *line*, int *offset*) [protected]

try to find a token on a string, begenning on given offset

Parameters

line	QString source
offset	index where to begin the search

Returns

Token

4.6.2.3 QList< Token > Lexer::run ( )

tokenize the source

Returns

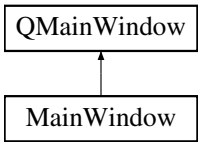
List of tokens

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/lexer.h
- /home/bob/dev/utt-nf05-project/sources/lib/lexer.cpp

4.7 MainWindow Class Reference

Inheritance diagram for MainWindow:



Public Member Functions

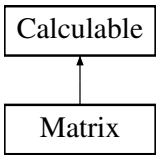
- **MainWindow** (QWidget *parent=0)
- virtual bool **event** (QEvent *event)

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/mainwindow.hpp
- /home/bob/dev/utt-nf05-project/sources/mainwindow.cpp

4.8 Matrix Class Reference

Inheritance diagram for Matrix:



Public Member Functions

- Matrix (QString value)

  constructor
- Matrix (Matrix &value)

  Copy constructor.
- Matrix ()

  Construct a matrix of one per one with the value 0.
- QString getValue ()

  value accessor
- QVector< QVector< double > > getRawValue ()

  Raw value accessor.
- void setValue (QString newValue)

  value setter
- void setRawValue (QVector< QVector< double > > newValue)

  raw value setter
- void setValue (Matrix *newValue)

  Copy setter.
- Calculable * operator* (Calculable &a)

  *overload operator * between two Calculable*
- Calculable * operator+ (Calculable &a)

  overload operator - between two Calculable
- Calculable * operator- (Calculable &a)

  overload operator + between two Calculable
- int getM ()

  Get the number of rows of the matrix.
- int getN ()

  Get the number of columns of the matrix.
- void setM (int M)

  Set the number of rows of the matrix.

- void setN (int N)

  Set the number of columns of the matrix.
- double getCell (const int i, const int j)

  Get the raw value of cellule.
- void setCell (const int i, const int j, const double value)

  Set the raw value of cellule.
- std::string getTypeStr ()

  Define the type of the element as a string.
- TokenKind getType ()

  Define the type of the element as a TokenKind from token.h.

Protected Attributes

- QVector< QVector< double > > value

  Handle the matrix raw value.
- int M

  Give the number of rows of the matrix.
- int N

  Give the number of columns of the matrix.

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Matrix::Matrix (QString *value*)

constructor

Parameters

value	- the string value

4.8.1.2 Matrix::Matrix (**Matrix** & *value*)

Copy constructor.

Parameters

value	- the matrix

4.8.2 Member Function Documentation

4.8.2.1 double Matrix::getCell (const int *i*, const int *j*)

Get the raw value of cellule.

Parameters

i	- The row of the cell between 0 and M-1
j	- The row of the cell between 0 and N-1

Returns

The raw value of the cellule

4.8.2.2 int Matrix::getM ( )

Get the number of rows of the matrix.

Returns

The number of rows

4.8.2.3 int Matrix::getN ( )

Get the number of columns of the matrix.

Returns

The number of columns

4.8.2.4 QVector< QVector< double > > Matrix::getRawValue ( )

Raw value accessor.

Returns

the raw value of the Calculable

4.8.2.5 QString Matrix::getValue ( ) [virtual]

value accessor

Returns

the value of the Calculable

Reimplemented from Calculable.

4.8.2.6 Calculable * Matrix::operator* (Calculable & *a*) [virtual]

overload operator * between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.8.2.7 Calculable * Matrix::operator+ (Calculable & *a*) [virtual]

overload operator - between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.8.2.8 Calculable * Matrix::operator- (Calculable & *a*) [virtual]

overload operator + between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.8.2.9 void Matrix::setCell (const int *i*, const int *j*, const double *value*)

Set the raw value of cellule.

Parameters

i	- The row of the cell between 0 and M-1
j	- The row of the cell between 0 and N-1
value	- The new raw value of the cellule

4.8.2.10 void Matrix::setM (int *M*)

Set the number of rows of the matrix.

Parameters

The	new number of rows.

4.8.2.11 void Matrix::setN (int *N*)

Set the number of columns of the matrix.

Parameters

The	new number of columns.

4.8.2.12 void Matrix::setRawValue (QVector< QVector< double > > *newValue*)

raw value setter

Parameters

newValue	the value to set

4.8.2.13 void Matrix::setValue (QString *newValue*) [virtual]

value setter

Parameters

newValue	the value to set

Reimplemented from Calculable.

4.8.2.14 void Matrix::setValue (Matrix * *newValue*)

Copy setter.

Parameters

newValue	the value to set

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/calculables/matrix.h
- /home/bob/dev/utt-nf05-project/sources/lib/calculables/matrix.cpp

4.9 MatrixLib Class Reference

Static Public Member Functions

- static Matrix * identity (int n)
  Create an identity matrix.
- static Scalar * cofactor (Matrix *source, int i, int j)
  Calculate matrix cofacteur.
- static Scalar * determinant (Matrix *source)
  Calculate determinant of a matrix.
- static Matrix * transpose (Matrix *source)
  Transpose a matrix.
- static Matrix * coMatrix (Matrix *source)
  Find cofactor matrix of source.
- static Matrix * inv (Matrix *source)
  Find the inverse matrix of source if possible, rise excepetion if not possible.
- static double trace (Matrix *source)
  Calculate trace of the matrix.
- static double norm (Matrix *source)
  Calculate norm of vertical or horizontal vector.

4.9.1 Member Function Documentation

4.9.1.1 Scalar * MatrixLib::cofactor (Matrix * *source*, int *i*, int *j*) [static]

Calculate matrix cofacteur.

Parameters

source	- The source matrix
i	- The line where the cofacteur is calculated (between 0 and M-1)
j	- The column where the cofacteur is calculated (between 0 and N-1)

Returns

The cofacteur

4.9.1.2 Matrix * MatrixLib::coMatrix (Matrix * *source*) [static]

Find cofactor matrix of source.

Parameters

source	- Source matrix

Returns

The cofactor matrix

4.9.1.3 Scalar * MatrixLib::determinant (Matrix * *source*) [static]

Calculate determinant of a matrix.

Parameters

source	- The source matrix

Returns

The determinant

4.9.1.4 Matrix * MatrixLib::identity (int *n*) [static]

Create an identity matrix.

Parameters

n	- Size of the matrix n*n

Returns

the identity matrix generated

4.9.1.5 Matrix * MatrixLib::inv (Matrix * *source*) [static]

Find the inverse matrix of source if possible, rise excepetion if not possible.

Parameters

source	- Source matrix

Returns

The inverse matrix

4.9.1.6 double MatrixLib::norm (Matrix * *source*) [static]

Calculate norm of vertical or horizontal vector.

Parameters

source	- Source column or line matrix

Returns

The norm

4.9.1.7 double MatrixLib::trace (Matrix * *source*) [static]

Calculate trace of the matrix.

Parameters

source	- Source matrix

Returns

The trace

4.9.1.8 Matrix * MatrixLib::transpose (Matrix * *source*) [static]

Transpose a matrix.

Parameters

source	- The source matrix

Returns

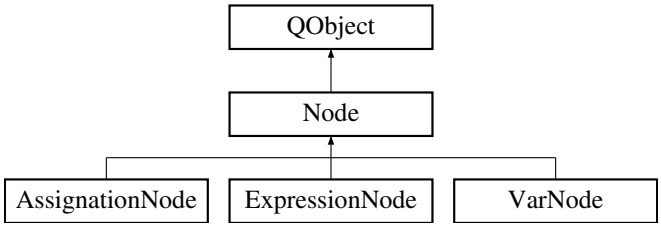
The transposed matrix

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/matrixlib.h
- /home/bob/dev/utt-nf05-project/sources/lib/matrixlib.cpp

4.10 Node Class Reference

Inheritance diagram for Node:



Public Member Functions

- Node ()
  constructor
- ~Node ()
  destructor
- virtual Calculable * execute ()=0
  Pure virtual method. Execute the node depending on its type (use polymorphism)
- virtual QString toString () const =0
  Pur virtual method. Return a string-based representation of the node.

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/node.h
- /home/bob/dev/utt-nf05-project/sources/lib/node.cpp

4.11 Operator Class Reference

Public Member Functions

- Operator (int precedence, Associativity associativity)
  constructor
- ~Operator ()
  destructor
- int getPrecedence ()
  predecence accessor
- Associativity getAssociativity ()
  associativity accessor

Static Public Member Functions

- static bool isOperator (Token token)
  determine if a token is an operator or a function

Public Attributes

- int **precedence**
- Associativity **associativity**

Static Public Attributes

- static QMap< int, Operator * > **operators** = Operator::initializeOperators()

Static Protected Member Functions

- static QMap< int, Operator * > initializeOperators ()
  fill the operators QMap

4.11.1 Constructor & Destructor Documentation

4.11.1.1 Operator::Operator (int *precedence,* Associativity *associativity*)

constructor

Parameters

precedence	precedence of the operator
associativity	associativity type of the operator

4.11.2 Member Function Documentation

4.11.2.1 Associativity Operator::getAssociativity ( )

associativity accessor

Returns

associativity type

4.11.2.2 int Operator::getPrecedence ( )

predecence accessor

Returns

predecence

4.11.2.3 QMap< int, Operator * > Operator::initializeOperators ( ) [static], [protected]

fill the operators QMap

Returns

a filled QMap containing all operators

4.11.2.4 bool Operator::isOperator (Token *token*) [static]

determine if a token is an operator or a function

Parameters

token	token to test

Returns

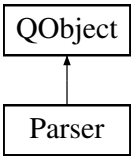
nature of the token

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/operator.h
- /home/bob/dev/utt-nf05-project/sources/lib/operator.cpp

4.12 Parser Class Reference

Inheritance diagram for Parser:



Signals

- void **parenthesisError** ()

Public Member Functions

- Parser (QString source)

  constructor
- ~Parser ()

  destructor
- Calculable * run ()

  run the parser

Static Public Member Functions

- static bool isFunction (Token token)

  isFunction check if a token is a function

4.12.1 Constructor & Destructor Documentation

4.12.1.1 Parser::Parser (QString *source*)

constructor

Parameters

source	QString to parse

4.12.2 Member Function Documentation

4.12.2.1 bool Parser::isFunction (Token *token*) [static]

isFunction check if a token is a function

Parameters

token	to test

Returns

type of token

4.12.2.2 Calculable * Parser::run ( )

run the parser

Returns

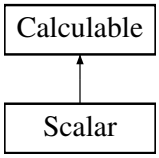
a Calculable

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/parser.h
- /home/bob/dev/utt-nf05-project/sources/lib/parser.cpp

4.13 Scalar Class Reference

Inheritance diagram for Scalar:



Public Member Functions

- Scalar (QString value)
  constructor
- Scalar (double value)
  constructor
- Scalar (Scalar &value)
  Copy constructor.
- QString getValue ()
  value accessor
- double getRawValue ()
  value accessor
- void setValue (QString newValue)
  value setter
- void setRawValue (double newValue)
  value setter
- Calculable * operator* (Calculable &a)
  *overload operator * between two Calculable*
- Calculable * operator/ (Calculable &a)
  overload operator / between two Calculable
- Calculable * operator+ (Calculable &a)
  overload operator + between two Calculable
- Calculable * operator- (Calculable &a)
  overload operator - between two Calculable
- Calculable * operator% (Calculable &a)
  overload operator % between two Calculable
- Calculable * operator^ (Calculable &a)
  overload operator ^ between two Calculable

- std::string getTypeStr ()

  Define the type of the element as a string.
- TokenKind getType ()

  Define the type of the element as a TokenKind from token.h.

Protected Attributes

- double **value**

4.13.1 Constructor & Destructor Documentation

4.13.1.1 Scalar::Scalar (QString *value*)

constructor

Parameters

value	- the string value

4.13.1.2 Scalar::Scalar (double *value*)

constructor

Parameters

value	the raw value

4.13.1.3 Scalar::Scalar (Scalar & *value*)

Copy constructor.

Parameters

value	the raw value

4.13.2 Member Function Documentation

4.13.2.1 double Scalar::getRawValue ( )

value accessor

Returns

the raw value of the Calculable

4.13.2.2 QString Scalar::getValue ( ) [virtual]

value accessor

Returns

the value of the Calculable

Reimplemented from Calculable.

4.13.2.3 Calculable * Scalar::operator% (Calculable & *a*) [virtual]

overload operator % between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.13.2.4 Calculable * Scalar::operator* (Calculable & *a*) [virtual]

overload operator * between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.13.2.5 Calculable * Scalar::operator+ (Calculable & *a*) [virtual]

overload operator + between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.13.2.6 Calculable * Scalar::operator- (Calculable & *a*) [virtual]

overload operator - between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.13.2.7 Calculable * Scalar::operator/ (Calculable & *a*) [virtual]

overload operator / between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.13.2.8 Calculable * Scalar::operator^ (Calculable & *a*) [virtual]

overload operator ^ between two Calculable

Parameters

a	a Calculable

Returns

a Calculable

Reimplemented from Calculable.

4.13.2.9 void Scalar::setRawValue (double *newValue*)

value setter

Parameters

newValue	the value to set

4.13.2.10 void Scalar::setValue (QString *newValue*) [virtual]

value setter

Parameters

newValue	the value to set

Reimplemented from Calculable.

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/calculables/scalar.h
- /home/bob/dev/utt-nf05-project/sources/lib/calculables/scalar.cpp

4.14 Token Class Reference

Public Member Functions

- Token (TokenKind kind, QString value="")

  constructor
- ~Token ()

  destructor
- TokenKind getKind () const

kind accessor

- QString getValue () const

  value accessor

- void setValue (QString value)

  setValue value setter

- void setKind (TokenKind kind)

  setKind kind setter

Protected Attributes

- TokenKind **kind**
- QString **value**

4.14.1 Constructor & Destructor Documentation

4.14.1.1 Token::Token (TokenKind *kind*, QString *value* = " ")

constructor

Parameters

kind	kind of token
value	value of token

4.14.2 Member Function Documentation

4.14.2.1 TokenKind Token::getKind ( ) const

kind accessor

Returns

kind of token

4.14.2.2 QString Token::getValue ( ) const

value accessor

Returns

value of token

4.14.2.3 void Token::setKind (TokenKind *kind*)

setKind kind setter

Parameters

kind	kind to set

4.14.2.4 void Token::setValue (QString *value*)

setValue value setter

Parameters

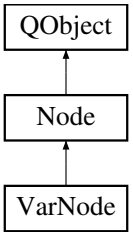
value	value to set

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/token.h
- /home/bob/dev/utt-nf05-project/sources/lib/token.cpp

4.15 VarNode Class Reference

Inheritance diagram for VarNode:



Public Member Functions

- VarNode (QString varName, Calculable ∗value)

  constructor
- ∼VarNode ()

  destructor
- virtual Calculable ∗ execute ()

  Execute the node.
- virtual QString toString () const

  toString method
- void setValue (Calculable ∗value)

  Set a value in the registry.
- Calculable ∗ getValue () const

  value accessor
- QString getName () const

  varName accessor

Static Public Member Functions

- static VarNode ∗ getVar (QString reference)

  return a var if it exists or a new one if it doesn't
- static QList< VarNode ∗ > ∗ getRegistry ()

  global registry accessor

Protected Attributes

- QString **varName**
- Calculable ∗ **value**

4.15.1 Constructor & Destructor Documentation

4.15.1.1 VarNode::VarNode (QString *varName,* Calculable * *value*)

constructor

Parameters

varName	QString of the varName
value	value

4.15.2 Member Function Documentation

4.15.2.1 Calculable * VarNode::execute ( ) [virtual]

Execute the node.

Returns

a Calculable pointer

Implements Node.

4.15.2.2 QString VarNode::getName ( ) const

varName accessor

Returns

the var name

4.15.2.3 QList< VarNode * > * VarNode::getRegistry ( ) [static]

global registry accessor

Returns

the global registry

4.15.2.4 Calculable * VarNode::getValue ( ) const

value accessor

Returns

a pointer to the node's value

4.15.2.5 VarNode * VarNode::getVar (QString *reference*) [static]

return a var if it exists or a new one if it doesn't

Parameters

reference	var name
registry	memory where to find/create the var

Returns

a pointer to the new/already existing VarNode

4.15.2.6 void VarNode::setValue (Calculable * *value*)

Set a value in the registry.

Parameters

value	Calculable to set

4.15.2.7 QString VarNode::toString ( ) const [virtual]

toString method

Returns

a QString representation of the Node

Implements Node.

The documentation for this class was generated from the following files:

- /home/bob/dev/utt-nf05-project/sources/lib/varNode.h
- /home/bob/dev/utt-nf05-project/sources/lib/varNode.cpp

Index