The Farsyte Toolkit

Generated by Doxygen 1.8.7

Mon Sep 15 2014 22:23:01

Contents

1 Main Page

Documentation for the publicly exported interfaces to the libraries provided by the Farsyte Toolkit.

A Note on Namespaces

All interfaces live within the `Farsyte` namespace, to avoid collisions with simiarly named things defined elsewhere.

Header File Overview

Farsyte::Utility "utility.h"

The `utility.h` header collects general utility code that is not associated with any of the larger packages within the toolkit project. This could be code that is shared across multiple packages (without an obvious owner), or it could be entire packages that are simply too small to warrant tracking as their own library.

Farsyte::Testing "testing.h"

The `testing.h` header provides declarations for the API for the library that is used to record test results in a form that allows an automated build system (currently the Bamboo continuous intergration system from Atlassian) to track testing results.

The `Testing` Library implements four classes:

- The Oops class is the base class for exceptions thrown within the Testing library representing failures of the library or failure to follow the requirements of the library when calling it.
- The Test class represents a single test, counting as one test when presenting counts of tests passed or failed. Tests may indicate one or more conditions that fail or were skipped, with supporting text, as well as errors occurring in the testing process. It is not uncommon for a Test to correspond to a single method name within a class, and for the code to test various aspects of the method in sequence. If two Test objects have the same test name and the same suite name, their results will be combined during reporting.
- The Suite class represents a collection of Test objects that are logically related; tests with the same Suite name are grouped together when reporting summaries of test results. It is not uncommon for a Suite to correspond to a single Class to be tested within a library. If two Suite objects have the same name, their results will be combined during reporting (even if they come from tests in quite different subprojects). Risks associated with this can be mitigated by assuring that class names are distinct between subprojects of a superproject, when tests are run across the entire superproject as a unit.
- The Log class corresponds to a single stream of test output produced by sequentially running a series test suites, each of which sequentially runs a series of test cases. Note that the Log class allows test programs to have multiple Log objects open at once, allowing a test program to persue multiple test series in parallel if this is appropriate and useful.

Farsyte::Matrix "matrix.h"

Matrix math is provided as a genearlized template for dense rectangular matrices, a derived template for working with column vectors, and a class representing a position in a three dimensional state space (such as distance north, east, and above the flagpole).

The Matrix Library implements two templates and one class:

- The Matrix class template supports dense rectangular matrices with any small positive integer number of rows and columns, containing data elements of any data type that supports the desired operations. This class currently provides class methods retrieving the dimensions of the matrix, and member methods providing the operations listed here. More operations will be added once I am happy with code quality, test coverage, and documentation for these operations:
  - Initialization
  - Addition
  - Difference
  - Negation
  - Transpose
- The ColVec class template is a proxy class providing the obvious API adjustments to use for column vectors, which are simply a one-column matrix. The API differences mainly involve not having to redundantly specify column numbers when working with a ColVec class.
- The ThreeVec class likewise provides for a three-element column vector such as might be used to represent a Position (north, east and above the flagpole), Velocity, Direction, and so on. The ThreeVec class adds the Cross Product operation, which is not provided for general Matrix and Column Vector objects.

2 Hierarchical Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

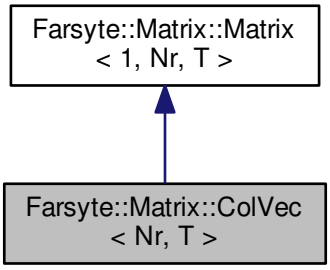
5 Class Documentation

5.1 Farsyte::Matrix::ColVec< Nr, T > Class Template Reference

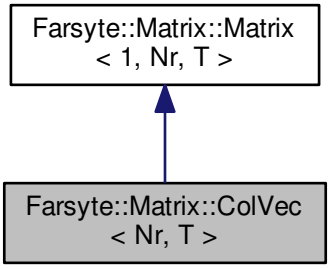
Column Vector Template.

`#include <matrix.h>`

Inheritance diagram for Farsyte::Matrix::ColVec< Nr, T >:



Collaboration diagram for Farsyte::Matrix::ColVec< Nr, T >:



Public Types

- typedef Matrix< 1, Nr, T > MatMe

  Typedef for matrix generalization.
- typedef MatMe::A A

  Typedef for array containing the data.
- typedef MatMe::C C

  Typedef for array containing one column of data.

Public Member Functions

- ColVec (A const &a)

  Construct a Column Vector from a 2-D Array.
- ColVec (C const &a)

  Construct a Column Vector from a 1-D Array.
- ColVec (MatMe const &c)

  Duplicate an existing Column Vector.
- T const & operator() (int ri) const

  ColVec Subscripting Operator.
- T & operator() (int ri)

  ColVec Subscripting Operator.

Protected Member Functions

- T const & sub (int ri) const

  ColVec Subscripting Implementation.
- T & sub (int ri)

  ColVec Subscripting Implementation.

Additional Inherited Members

5.1.1 Detailed Description

template<int Nr, typename T>class Farsyte::Matrix::ColVec< Nr, T >

Column Vector Template.

Parameters

Nr	Number of rows in the vector.
T	Data type for each vector element.

This template implements a column vector of a compile-time determined size, containing elements of compile-time determined type.

Definition at line 390 of file matrix.h.

5.1.2 Member Typedef Documentation

5.1.2.1 template<int Nr, typename T> typedef Matrix<1,Nr,T> Farsyte::Matrix::ColVec< Nr, T >::MatMe

Typedef for matrix generalization.

Definition at line 397 of file matrix.h.

5.1.3 Constructor & Destructor Documentation

5.1.3.1 template<int Nr, typename T> Farsyte::Matrix::ColVec< Nr, T >::ColVec (A const & *a*) [inline]

Construct a Column Vector from a 2-D Array.

Parameters

a	An appropriately shaped object of the C++ array template type. This initializes the Column Vector data to contain values from the corresponding elements of the array provided.

Definition at line 441 of file matrix.h.

```
00442         : MatMe(a)
00443         {
00444         }
```

5.1.3.2 template<int Nr, typename T> Farsyte::Matrix::ColVec< Nr, T >::ColVec (C const & *a*) [inline]

Construct a Column Vector from a 1-D Array.

Parameters

a	An appropriately shaped object of the C++ array template type. This initializes the Column Vector data to contain values from the corresponding elements of the array provided.

Definition at line 451 of file matrix.h.

```
00452         : MatMe(A{{a}})
00453         {
00454         }
```

5.1.3.3 template<int Nr, typename T> Farsyte::Matrix::ColVec< Nr, T >::ColVec (MatMe const & *c*) [inline]

Duplicate an existing Column Vector.

Parameters

c	A column vector to duplicate. Initialize this column vector to be a duplicate of the one provided.

Note

Can be called with any appropriately dimensioned matrix.

Definition at line 461 of file matrix.h.

```
00462         : MatMe(c)
00463         {
00464         }
```

5.1.4 Member Function Documentation

5.1.4.1 template<int Nr, typename T> T const& Farsyte::Matrix::ColVec< Nr, T >::operator() (int *ri*) const [inline]

ColVec Subscripting Operator.

Parameters

ri	Row Index, in the range 1 to N inclusive.

Returns

read-only reference to the selected element.

Note

Fortran conventions for array subscripting.

Definition at line 471 of file matrix.h.

```
00472         {
00473           return sub(ri);
00474         }
```

5.1.4.2 template<int Nr, typename T> T& Farsyte::Matrix::ColVec< Nr, T >::operator() (int *ri*) [inline]

ColVec Subscripting Operator.

Parameters

ri	Row Index, in the range 1 to N inclusive.

Returns

modifiable reference to the selected element.

Note

Fortran conventions for array subscripting.

Definition at line 481 of file matrix.h.

```
00482         {
00483           return sub(ri);
00484         }
```

5.1.4.3 template<int Nr, typename T> T const& Farsyte::Matrix::ColVec< Nr, T >::sub (int *ri*) const [inline], [protected]

ColVec Subscripting Implementation.

Parameters

ri	Row Index, in the range 1 to N inclusive.

Returns

read-only reference to the selected element.

Note

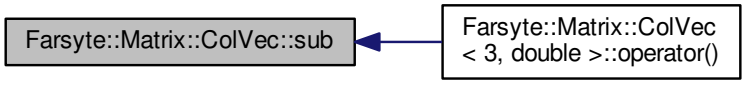
Fortran conventions for array subscripting.

Definition at line 414 of file matrix.h.

Referenced by Farsyte::Matrix::ColVec< 3, double >::operator()().

```
00415         {
00416           return MatMe::sub(ri, 1);
00417         }
```

Here is the caller graph for this function:



5.1.4.4 template<int Nr, typename T> T& Farsyte::Matrix::ColVec< Nr, T >::sub (int *ri*) [inline], [protected]

ColVec Subscripting Implementation.

Parameters

ri	Row Index, in the range 1 to N inclusive.

Returns

modifiable reference to the selected element.

Note

Fortran conventions for array subscripting.

Definition at line 424 of file matrix.h.

```
00425         {
00426           return MatMe::sub(ri, 1);
00427         }
```

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

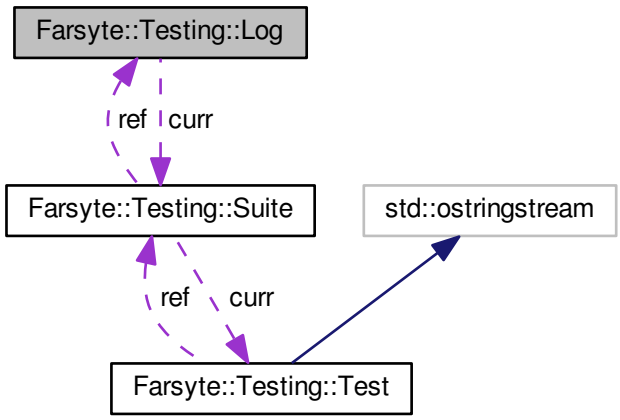
- matrix.h

5.2 Farsyte::Testing::Log Class Reference

The Log Object.

`#include <testing.h>`

Collaboration diagram for Farsyte::Testing::Log:



Public Member Functions

- Log (std::ostream &out, std::string const &name)

  Log Constructor.
- ~Log ()

  Log Destructor.

Public Attributes

- std::ostream & out

  Current output stream.
- std::string const name

  Name of the test log.
- Suite * curr

  Currently active Suite, or NULL if none.
- size_t suites

  Cumulative count of Suites.
- size_t tests

  Cumulative count of Tests.
- size_t failed_tests

  Cumulative count of Tests with at least one FAIL.
- size_t skipped_tests

  Cumulative count of Tests with at least one SKIP.
- size_t errored_tests

  Cumulative count of Tests with at least one ERROR.
- size_t total_fails

Cumulative count of all FAIL reports.

- size_t total_skips

  Cumulative count of all SKIP reports.

- size_t total_errors

  Cumulative count of all ERROR reports.

5.2.1 Detailed Description

The Log Object.

The Log object constructor writes XML text to the specified output stream to start a new log file. The destructor writes trailer data to finish the XML element opened at the top. Test programs create a Log object for each file they want to write, and may have several Log file objects open at the same time.

Definition at line 33 of file testing.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Farsyte::Testing::Log::Log (std::ostream & *out*, std::string const & *name*)

Log Constructor.

Parameters

out	- where to send the XML text.
name	- the name of the Log. Writes XML header text to the output stream.

5.2.2.2 Farsyte::Testing::Log::~Log ( )

Log Destructor.

Writes XML trailer text to the output stream.

5.2.3 Member Data Documentation

5.2.3.1 Suite* Farsyte::Testing::Log::curr

Currently active Suite, or NULL if none.

Definition at line 43 of file testing.h.

5.2.3.2 size_t Farsyte::Testing::Log::errored_tests

Cumulative count of Tests with at least one ERROR.

Definition at line 58 of file testing.h.

5.2.3.3 size_t Farsyte::Testing::Log::failed_tests

Cumulative count of Tests with at least one FAIL.

Definition at line 52 of file testing.h.

5.2.3.4 std::ostream& Farsyte::Testing::Log::out

Current output stream.

Definition at line 37 of file testing.h.

5.2.3.5 size_t Farsyte::Testing::Log::skipped_tests

Cumulative count of Tests with at least one SKIP.

Definition at line 55 of file testing.h.

5.2.3.6 size_t Farsyte::Testing::Log::suites

Cumulative count of Suites.

Definition at line 46 of file testing.h.

5.2.3.7 size_t Farsyte::Testing::Log::tests

Cumulative count of Tests.

Definition at line 49 of file testing.h.

5.2.3.8 size_t Farsyte::Testing::Log::total_errors

Cumulative count of all ERROR reports.

Definition at line 67 of file testing.h.

5.2.3.9 size_t Farsyte::Testing::Log::total_fails

Cumulative count of all FAIL reports.

Definition at line 61 of file testing.h.

5.2.3.10 size_t Farsyte::Testing::Log::total_skips

Cumulative count of all SKIP reports.

Definition at line 64 of file testing.h.

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

- testing.h

5.3 Farsyte::Matrix::Matrix< Nc, Nr, T > Class Template Reference

Matrix Template.

`#include <matrix.h>`

Public Types

- typedef T value_type

  Typedef for type of matrix elements.
- typedef T & reference

  Reference to a matrix element.
- typedef T const & const_reference

  Const reference to a matrix element.
- typedef T * pointer

Pointer to a matrix element.

- typedef T const * const_pointer

  Const pointer to a matrix element.
- typedef std::array< T, Nr > C

  Typedef for array containing one column of the data.
- typedef std::array< C, Nc > A

  Typedef for array containing the data.

Public Member Functions

- Matrix ()

  Matrix Default Constructor.
- Matrix (T const &d)

  Matrix Diagonal Constructor.
- Matrix (A const &a)

  Matrix Construction from Array of Arrays.
- Matrix (Matrix const &m)

  Duplicate Matrix Construction.
- T const & operator() (int ri, int ci) const

  Matrix Subscripting Operator.
- T & operator() (int ri, int ci)

  Matrix Subscripting Operator.
- bool equals (Matrix const &p) const

  Matrix Equality Test.
- Matrix & increment_by (Matrix const &p)

  Matrix Increment operation.
- Matrix & operator+= (Matrix const &p)

  Matrix Increment operator.
- Matrix & decrement_by (Matrix const &p)

  Matrix Decrement operation.
- Matrix & operator-= (Matrix const &p)

  Matrix Decrement operator.
- Matrix negate ()

  Matrix Negate operation.
- Matrix< Nr, Nc, T > transpose () const

  Matrix Transpose operation.

Static Public Member Functions

- static size_t rows ()

  Matrix rows.
- static size_t cols ()

  Matrix columns.
- static size_t size ()

  Matrix elements.

Protected Member Functions

- T const & sub (int ri, int ci) const

  Matrix Subscripting Implementation.
- T & sub (int ri, int ci)

  Matrix Subscripting Implementation.

Protected Attributes

- A data

  Storage for Matrix State.

5.3.1 Detailed Description

template<int Nc, int Nr, typename T>class Farsyte::Matrix::Matrix< Nc, Nr, T >

Matrix Template.

Parameters

Nc	Number of columns in the matrix.
Nr	Number of rows in the matrix.
T	Data type for each matrix element.

This template implements rectangular matrices of a compile-time determined size, containing elements of compile-time determined type.

Definition at line 37 of file matrix.h.

5.3.2 Member Typedef Documentation

5.3.2.1 template<int Nc, int Nr, typename T> typedef T const* Farsyte::Matrix::Matrix< Nc, Nr, T >::const_pointer

Const pointer to a matrix element.

Definition at line 54 of file matrix.h.

5.3.2.2 template<int Nc, int Nr, typename T> typedef T const& Farsyte::Matrix::Matrix< Nc, Nr, T >::const_reference

Const reference to a matrix element.

Definition at line 48 of file matrix.h.

5.3.2.3 template<int Nc, int Nr, typename T> typedef T* Farsyte::Matrix::Matrix< Nc, Nr, T >::pointer

Pointer to a matrix element.

Definition at line 51 of file matrix.h.

5.3.2.4 template<int Nc, int Nr, typename T> typedef T& Farsyte::Matrix::Matrix< Nc, Nr, T >::reference

Reference to a matrix element.

Definition at line 45 of file matrix.h.

5.3.2.5 template<int Nc, int Nr, typename T> typedef T Farsyte::Matrix::Matrix< Nc, Nr, T >::value_type

Typedef for type of matrix elements.

Definition at line 42 of file matrix.h.

5.3.3 Constructor & Destructor Documentation

5.3.3.1 template<int Nc, int Nr, typename T> Farsyte::Matrix::Matrix< Nc, Nr, T >::Matrix ( ) [inline]

Matrix Default Constructor.

Matrix objects that are default-constructed are assured of having each element appropriately initialized.

Definition at line 126 of file matrix.h.

```
00127         : data()
00128         {
00129         }
```

5.3.3.2 template<int Nc, int Nr, typename T> Farsyte::Matrix::Matrix< Nc, Nr, T >::Matrix (T const & *d*) [inline]

Matrix Diagonal Constructor.

Parameters

d	Value to copy into each diagonal element.

Definition at line 134 of file matrix.h.

```
00135         : data()
00136         {
00137           for (size_t i = 1; (i <= Nr) && (i <= Nc); ++i)
00138             sub(i,i) = d;
00139         }
```

5.3.3.3 template<int Nc, int Nr, typename T> Farsyte::Matrix::Matrix< Nc, Nr, T >::Matrix (A const & *a*) [inline]

Matrix Construction from Array of Arrays.

Parameters

a	Array to duplicate. This method is used by subclasses to provide value construction of Matrices using Arrays of Arrays of the appropriate dimensions.

Note

Not a public interface: only classes within the class heirarchy below Matrix should be aware of the data organization within the Matrix object.

Definition at line 150 of file matrix.h.

```
00151         : data(a)
00152         {
00153         }
```

5.3.3.4 template<int Nc, int Nr, typename T> Farsyte::Matrix::Matrix< Nc, Nr, T >::Matrix (Matrix< Nc, Nr, T > const & *m*) [inline]

Duplicate Matrix Construction.

Parameters

m	Matrix to duplicate. Initialize this matrix to duplicate the data contained in the provided matrix.

Definition at line 160 of file matrix.h.

```
00161         : data(m.data)
00162         {
00163         }
```

5.3.4 Member Function Documentation

5.3.4.1 template<int Nc, int Nr, typename T> static size_t Farsyte::Matrix::Matrix< Nc, Nr, T >::cols ( ) [inline], [static]

Matrix columns.

Returns

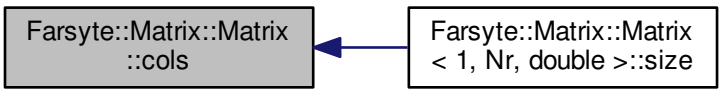
number of columns `Nc` in tha matrix.

Definition at line 75 of file matrix.h.

Referenced by Farsyte::Matrix::Matrix< 1, Nr, double >::size().

```
00076         {
00077           return Nc;
00078         }
```

Here is the caller graph for this function:



5.3.4.2 template<int Nc, int Nr, typename T> Matrix& Farsyte::Matrix::Matrix< Nc, Nr, T >::decrement_by (Matrix< Nc, Nr, T > const & *p*) [inline]

Matrix Decrement operation.

Parameters

p	Matrix of decrement values.

Returns

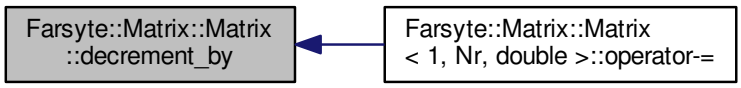
this matrix, after decrementing. Each element of this matrix is decremented by the value of the corresponding element of the provided matrix.

Definition at line 234 of file matrix.h.

Referenced by Farsyte::Matrix::Matrix< 1, Nr, double >::operator-=().

```
00235         {
00236           for (int ci = 1; ci <= Nc; ++ci)
00237             for (int ri = 1; ri <= Nr; ++ri)
00238               sub(ri,ci) -= p.sub(ri,ci);
00239           return *this;
00240         }
```

Here is the caller graph for this function:



5.3.4.3 template<int Nc, int Nr, typename T> bool Farsyte::Matrix::Matrix< Nc, Nr, T >::equals (Matrix< Nc, Nr, T > const & *p*) const [inline]

Matrix Equality Test.

Parameters

p	Matrix to compare.

Returns

true if all elements compare equal, else false.

Definition at line 191 of file matrix.h.

```
00192         {
00193           for (int ci = 1; ci <= Nc; ++ci)
00194             for (int ri = 1; ri <= Nr; ++ri)
00195               if (sub(ri,ci) != p.sub(ri,ci))
00196                 return false;
00197           return true;
00198         }
```

5.3.4.4 template<int Nc, int Nr, typename T> Matrix& Farsyte::Matrix::Matrix< Nc, Nr, T >::increment_by (Matrix< Nc, Nr, T > const & *p*) [inline]

Matrix Increment operation.

Parameters

p	Matrix of increment values.

Returns

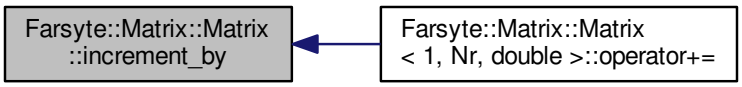
this matrix, after incrementing. Each element of this matrix is incremented by the value of the corresponding element of the provided matrix.

Definition at line 207 of file matrix.h.

Referenced by Farsyte::Matrix::Matrix< 1, Nr, double >::operator+=().

```
00208         {
00209           for (int ci = 1; ci <= Nc; ++ci)
00210             for (int ri = 1; ri <= Nr; ++ri)
00211               sub(ri,ci) += p.sub(ri,ci);
00212           return *this;
00213         }
```

Here is the caller graph for this function:



5.3.4.5 template<int Nc, int Nr, typename T> Matrix Farsyte::Matrix::Matrix< Nc, Nr, T >::negate ( ) [inline]

Matrix Negate operation.

Returns

self after negating elements.

Definition at line 257 of file matrix.h.

```
00258         {
00259           for (int ci = 1; ci <= Nc; ++ci)
00260             for (int ri = 1; ri <= Nr; ++ri)
00261               sub(ri,ci) = -sub(ri,ci);
00262           return *this;
00263         }
```

5.3.4.6 template<int Nc, int Nr, typename T> T const& Farsyte::Matrix::Matrix< Nc, Nr, T >::operator() (int *ri*, int *ci*) const [inline]

Matrix Subscripting Operator.

Parameters

ri	Row Index, in the range 1 to Nr inclusive.
ci	Column Index, in the range 1 to Nc inclusive.

Returns

read-only reference to the selected element.

Note

Fortran conventions for array subscripting.

Definition at line 171 of file matrix.h.

```
00172         {
00173           return sub(ri, ci);
00174         }
```

5.3.4.7 template<int Nc, int Nr, typename T> T& Farsyte::Matrix::Matrix< Nc, Nr, T >::operator() (int *ri*, int *ci*) [inline]

Matrix Subscripting Operator.

Parameters

ri	Row Index, in the range 1 to Nr inclusive.
ci	Column Index, in the range 1 to Nc inclusive.

Returns

modifiable reference to the selected element.

Note

Fortran conventions for array subscripting.

Definition at line 182 of file matrix.h.

```
00183         {
00184           return sub(ri, ci);
00185         }
```

5.3.4.8 template<int Nc, int Nr, typename T> Matrix& Farsyte::Matrix::Matrix< Nc, Nr, T >::operator+= (Matrix< Nc, Nr, T > const & *p*) [inline]

Matrix Increment operator.

Parameters

p	Matrix of increment values.

Returns

this matrix, after incrementing. Each element of this matrix is incremented by the value of the corresponding element of the provided matrix.

Definition at line 222 of file matrix.h.

```
00223         {
00224           return increment_by(p);
00225         }
```

5.3.4.9 template<int Nc, int Nr, typename T> Matrix& Farsyte::Matrix::Matrix< Nc, Nr, T >::operator-= (Matrix< Nc, Nr, T > const & *p*) [inline]

Matrix Decrement operator.

Parameters

p	Matrix of decrement values.

Returns

this matrix, after decrementing. Each element of this matrix is decremented by the value of the corresponding element of the provided matrix.

Definition at line 249 of file matrix.h.

```
00250         {
00251           return decrement_by(p);
00252         }
```

5.3.4.10 template<int Nc, int Nr, typename T> static size_t Farsyte::Matrix::Matrix< Nc, Nr, T >::rows ( ) [inline], [static]

Matrix rows.

Returns

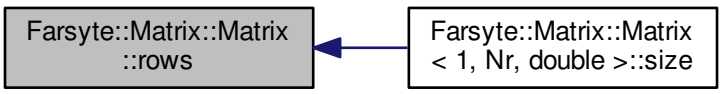
number of rows `Nr` in tha matrix.

Definition at line 67 of file matrix.h.

Referenced by Farsyte::Matrix::Matrix< 1, Nr, double >::size().

```
00068         {
00069           return Nr;
00070         }
```

Here is the caller graph for this function:



5.3.4.11 template<int Nc, int Nr, typename T> static size_t Farsyte::Matrix::Matrix< Nc, Nr, T >::size ( ) [inline], [static]

Matrix elements.

Returns

number of elements Nr*Nc in the matrix.

Definition at line 83 of file matrix.h.

```
00084         {
00085           return rows() * cols();
00086         }
```

5.3.4.12 template<int Nc, int Nr, typename T> T const& Farsyte::Matrix::Matrix< Nc, Nr, T >::sub (int *ri,* int *ci*) const [inline], [protected]

Matrix Subscripting Implementation.

Parameters

ri	Row Index, ranging from 1 to Nr inclusive.
ci	Column Index, ranging from 1 to Nc inclusive.

Returns

a read-only reference to the selected element.

Note

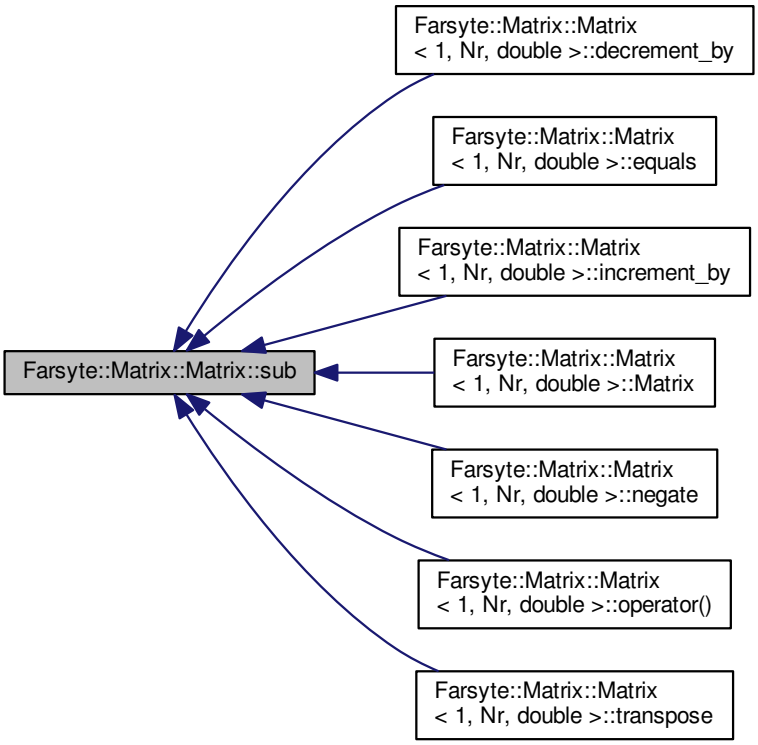
Fortran conventions for array subscripting.

Definition at line 96 of file matrix.h.

Referenced by Farsyte::Matrix::Matrix< 1, Nr, double >::decrement_by(), Farsyte::Matrix::Matrix< 1, Nr, double >::equals(), Farsyte::Matrix::Matrix< 1, Nr, double >::increment_by(), Farsyte::Matrix::Matrix< 1, Nr, double >::Matrix(), Farsyte::Matrix::Matrix< 1, Nr, double >::negate(), Farsyte::Matrix::Matrix< 1, Nr, double >::operator()(), and Farsyte::Matrix::Matrix< 1, Nr, double >::transpose().

```
00097         {
00098 #ifdef RANGE_CHECKER
00099           RANGE_CHECKER(1,ci,Nc);
00100           RANGE_CHECKER(1,ri,Nr);
00101 #endif
00102           return data[ci-1][ri-1];
00103         }
```

Here is the caller graph for this function:



5.3.4.13 template<int Nc, int Nr, typename T> T& Farsyte::Matrix::Matrix< Nc, Nr, T >::sub (int *ri,* int *ci*) [inline], [protected]

Matrix Subscripting Implementation.

Parameters

ri	Row Index, ranging from 1 to Nr inclusive.
ci	Column Index, ranging from 1 to Nc inclusive.

Returns

a writable reference to the selected element.

Note

Fortran conventions for array subscripting.

Definition at line 111 of file matrix.h.

```
00112         {
00113 #ifdef RANGE_CHECKER
```

```
00114           RANGE_CHECKER(1,ci,Nc);
00115           RANGE_CHECKER(1,ri,Nr);
00116 #endif
00117           return data[ci-1][ri-1];
00118         }
```

5.3.4.14 template<int Nc, int Nr, typename T> Matrix<Nr,Nc,T> Farsyte::Matrix::Matrix< Nc, Nr, T >::transpose ( ) const [inline]

Matrix Transpose operation.

Returns

transposed matrix.

Definition at line 269 of file matrix.h.

```
00270         {
00271           Matrix<Nr,Nc,T> R;
00272           for (int ci = 1; ci <= Nc; ++ci)
00273             for (int ri = 1; ri <= Nr; ++ri)
00274               R(ci,ri) = sub(ri,ci);
00275           return R;
00276         }
```

5.3.5 Member Data Documentation

5.3.5.1 template<int Nc, int Nr, typename T> A Farsyte::Matrix::Matrix< Nc, Nr, T >::data [protected]

Storage for Matrix State.

Definition at line 280 of file matrix.h.

Referenced by Farsyte::Matrix::Matrix< 1, Nr, double >::sub().

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

- matrix.h

5.4 Farsyte::Testing::Oops Class Reference

The Oops Object.

`#include <testing.h>`

Public Member Functions

- Oops (std::string f, int l, std::string c)

  Create a new Oops object.
- virtual std::ostream & print (std::ostream &s) const

  Print exception details to output stream.
- virtual void cancel () const

  Cancel exception printing.
- virtual ~Oops ()

  Object Destructor.

Public Attributes

- std::string file

  source file name.
- int line

  source line number.
- std::string cond

  one-line failed condition text
- bool pend

  true if not yet dealt with

5.4.1 Detailed Description

The Oops Object.

Base class for Exception heirarchy for the Testing library. Errors in the Testing library or in the way test code uses the testing library MAY be reported by throwing an object of a class derived from Oops.

Definition at line 258 of file testing.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Farsyte::Testing::Oops::Oops (std::string *f,* int *l,* std::string *c*)

Create a new Oops object.

Parameters

f	- source file name.
l	- source line number.
c	- condition that was violated.

5.4.2.2 virtual Farsyte::Testing::Oops::~Oops ( ) [virtual]

Object Destructor.

If the object has not been printed (or cancelled), prints the exception details to the standard error output. All resources owned by the object are released.

5.4.3 Member Function Documentation

5.4.3.1 virtual void Farsyte::Testing::Oops::cancel ( ) const [virtual]

Cancel exception printing.

This method simply marks the object as having been printed, without the overhead of actually printing the details.

5.4.3.2 virtual std::ostream& Farsyte::Testing::Oops::print (std::ostream & *s*) const [virtual]

Print exception details to output stream.

Parameters

s	what stream gets the output text.

Returns

the stream after sending the text. This method produces all available details from this exception object onto the specified output stream (and marks the object as having been printed).

5.4.4 Member Data Documentation

5.4.4.1 std::string Farsyte::Testing::Oops::file

source file name.

Definition at line 260 of file testing.h.

5.4.4.2 int Farsyte::Testing::Oops::line

source line number.

Definition at line 261 of file testing.h.

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

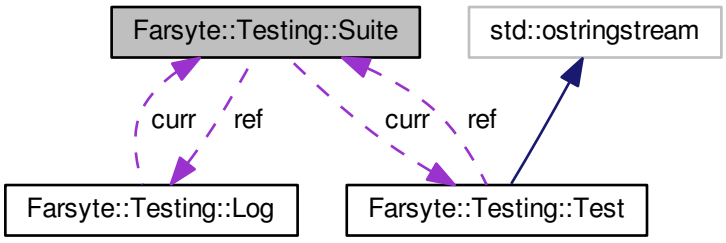
- testing.h

5.5 Farsyte::Testing::Suite Class Reference

The Suite Object.

`#include <testing.h>`

Collaboration diagram for Farsyte::Testing::Suite:



Public Member Functions

- Suite (Log &ref, std::string const &name)

  Construct a new Suite object.

- ~Suite ()

  Suite Destructor.

Public Attributes

- Log & ref

  Test Log associated with this Suite.
- std::string const name

  Name of the test suite.
- Test * curr

  Currently active Test, or NULL if none.
- size_t tests

  Count of tests within this suite.
- size_t failed_tests

  Cumulative count of Tests with at least one FAIL.
- size_t skipped_tests

  Cumulative count of Tests with at least one SKIP.
- size_t errored_tests

  Cumulative count of Tests with at least one ERROR.
- size_t total_fails

  Cumulative count of all FAIL reports.
- size_t total_skips

  Cumulative count of all SKIP reports.
- size_t total_errors

  Cumulative count of all ERROR reports.

5.5.1 Detailed Description

The Suite Object.

The Suite object constructor writes appropraite opening text to the XML output stream. The destructor writes appropriate text to the XML stream to close the XML element corresponding to the suite. It is an error to construct a Suite object for a Log that currently has an active Suite object; be sure that each Suite object goes out of scope before the next one is constructed.

Definition at line 92 of file testing.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Farsyte::Testing::Suite::Suite (Log & *ref*, std::string const & *name*)

Construct a new Suite object.

Parameters

ref	- Log that contains this Suite.

name	- what to call this Suite. Generates appropriate header text to the XML text output stream to start a new suite. May throw an exception if there is already an existing Suite associated with the specified Log.

5.5.2.2 Farsyte::Testing::Suite::~Suite ( )

Suite Destructor.

Generates an appropriate trailer to the XML text output stream to close the Suite. May throw an exception if it appears that this is not the current Suite associated with the Log.

5.5.3 Member Data Documentation

5.5.3.1 Test* Farsyte::Testing::Suite::curr

Currently active Test, or NULL if none.

Definition at line 102 of file testing.h.

5.5.3.2 size_t Farsyte::Testing::Suite::errored_tests

Cumulative count of Tests with at least one ERROR.

Definition at line 114 of file testing.h.

5.5.3.3 size_t Farsyte::Testing::Suite::failed_tests

Cumulative count of Tests with at least one FAIL.

Definition at line 108 of file testing.h.

5.5.3.4 Log& Farsyte::Testing::Suite::ref

Test Log associated with this Suite.

Definition at line 96 of file testing.h.

5.5.3.5 size_t Farsyte::Testing::Suite::skipped_tests

Cumulative count of Tests with at least one SKIP.

Definition at line 111 of file testing.h.

5.5.3.6 size_t Farsyte::Testing::Suite::tests

Count of tests within this suite.

Definition at line 105 of file testing.h.

5.5.3.7 size_t Farsyte::Testing::Suite::total_errors

Cumulative count of all ERROR reports.

Definition at line 123 of file testing.h.

5.5.3.8 size_t Farsyte::Testing::Suite::total_fails

Cumulative count of all FAIL reports.

Definition at line 117 of file testing.h.

5.5.3.9 size_t Farsyte::Testing::Suite::total_skips

Cumulative count of all SKIP reports.

Definition at line 120 of file testing.h.

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

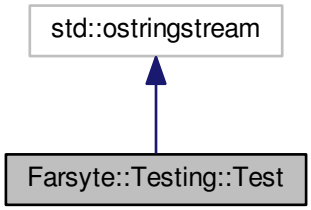
- testing.h

5.6 Farsyte::Testing::Test Class Reference

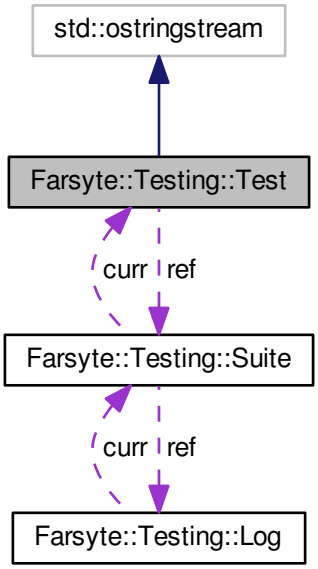
The Test object.

`#include <testing.h>`

Inheritance diagram for Farsyte::Testing::Test:



Collaboration diagram for Farsyte::Testing::Test:



Public Member Functions

- Test (Suite &ref, std::string const &name)

  Construct a new Test object.
- void fail (std::string const &cond)

  Register a Test FAIL.
- void skip (std::string const &cond)

  Register a Test SKIP.
- void error (std::string const &cond)

  Register a Test ERROR.
- void pass (std::string const &cond)

  Register a Test PASS.
- ~Test ()

  Suite Destructor.

Public Attributes

- Suite & ref

  Test Suite associated with this Test.
- std::string const name

  Name of the test case.
- size_t fails

number of FAIL reports for this test.

- size_t skips

  number of SKIP reports for this test.

- size_t errors

  number of ERROR reports for this test.

5.6.1 Detailed Description

The Test object.

Derived from Output String Stream.

The Test object constructor writes appropriate opening text to the XML file associatd with the Suite. The destructor writes appropriate text to the XML file to close the XML element corresponding to the test. Methods on Test objects are available for reporting test conditions that are skipped, test conditions that fail, and errors encountered during testing. It is an error to construct a Test object for a Suite that currently has an active Test object. Be sure that each Test object goes out of scope before the next one is constructed.

Data sent to this object using the << operator will be formatted appropriately and displayed by Bamboo as supporting text in appropriate reporting conditions.

Definition at line 163 of file testing.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Farsyte::Testing::Test::Test (Suite & *ref,* std::string const & *name*)

Construct a new Test object.

Parameters

ref	- Suite that contains this Test.
name	- what to call this Test. Generates appropriate header text to the XML text output stream to start a new Test. May throw an exception if there is already an existing Test associated with the specified Suite.

5.6.2.2 Farsyte::Testing::Test::~Test ( )

Suite Destructor.

Generates an appropriate trailer to the XML text output stream to close the Test. May throw an exception if it appears that this is not the current Test associated with the Suite.

Any text sent to the Test object using the C++ << operator that was not used in a FAIL, SKIP, or ERROR report will be attached to the test as supporting text.

5.6.3 Member Function Documentation

5.6.3.1 void Farsyte::Testing::Test::error (std::string const & *cond*)

Register a Test ERROR.

Parameters

cond	– one-line description of the error.

Sends text to the XML log indicating that a test has encountered a testing error, as described in the parameter string. Any accumulated text sent to the Test object with the C++ << operator will be included in the ERROR object as supporting text.

5.6.3.2 void Farsyte::Testing::Test::fail (std::string const & *cond*)

Register a Test FAIL.

Parameters

cond	– one-line description of failed condition.

Sends text to the XML log indicating that a test condition has failed, as described in the parameter string. Any accumulated text sent to the Test object with the C++ << operator will be included in the FAIL object as supporting text.

5.6.3.3 void Farsyte::Testing::Test::pass (std::string const & *cond*)

Register a Test PASS.

Parameters

cond	– one-line description of passed condition.

Mark that a test condition has PASSED. There is no XML output for Bamboo in this case. The purpose of this call is to correctly assocaite any supporting text sent to the Test object with this PASSING condition rather than including it in a subsequent fail, skip, or error.

5.6.3.4 void Farsyte::Testing::Test::skip (std::string const & *cond*)

Register a Test SKIP.

Parameters

cond	– one-line description of skipped condition.

Sends text to the XML log indicating that a test condition has been skipped, as described in the parameter string. Any accumulated text sent to the Test object with the C++ << operator will be included in the SKIP object as supporting text.

5.6.4 Member Data Documentation

5.6.4.1 size_t Farsyte::Testing::Test::errors

number of ERROR reports for this test.

Definition at line 180 of file testing.h.

5.6.4.2 size_t Farsyte::Testing::Test::fails

number of FAIL reports for this test.

Definition at line 174 of file testing.h.

5.6.4.3 Suite& Farsyte::Testing::Test::ref

Test Suite associated with this Test.

Definition at line 168 of file testing.h.

5.6.4.4 size_t Farsyte::Testing::Test::skips

number of SKIP reports for this test.

Definition at line 177 of file testing.h.

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

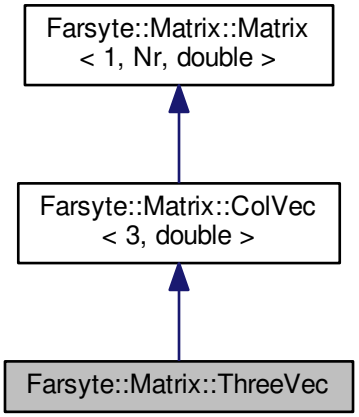
- testing.h

5.7 Farsyte::Matrix::ThreeVec Class Reference

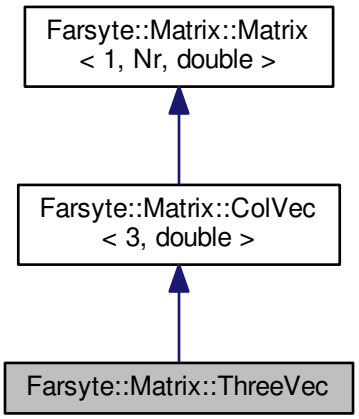
ThreeVec Class.

`#include <matrix.h>`

Inheritance diagram for Farsyte::Matrix::ThreeVec:



Collaboration diagram for Farsyte::Matrix::ThreeVec:



Public Member Functions

- ThreeVec ()

  ThreeVec Default Constructor.
- ThreeVec (double x, double y, double z)

  ThreeVec Constructor for Given Coefficients.
- ThreeVec (MatMe const &p)

  Duplicate Constructor for ThreeVec.

Protected Types

- typedef ColVec< 3, double > ColMe

  Typedef for ColVec generalization.
- typedef ColMe::MatMe MatMe

  Typedef for Matrix generalization.
- typedef ColMe::C C

  Typedef for 1-D array containing the data.

Additional Inherited Members

5.7.1 Detailed Description

ThreeVec Class.

This class implements a representation of a ThreeVec (more precisely a change in position), expressed as a column vector of three double precision components.

Definition at line 495 of file matrix.h.

5.7.2 Member Typedef Documentation

5.7.2.1 typedef ColMe::MatMe Farsyte::Matrix::ThreeVec::MatMe [protected]

Typedef for Matrix generalization.

Definition at line 503 of file matrix.h.

5.7.3 Constructor & Destructor Documentation

5.7.3.1 Farsyte::Matrix::ThreeVec::ThreeVec ( )

ThreeVec Default Constructor.

This class assures that all ThreeVec objects are initialized to zero when they are constructed, if no initial value is specified via a different constructor.

5.7.3.2 Farsyte::Matrix::ThreeVec::ThreeVec (double *x*, double *y*, double *z*)

ThreeVec Constructor for Given Coefficients.

Parameters

x	X coefficient for position.
y	Y coefficient for position.
z	Z coefficient for position. Initializes this position to contain the specified coefficients for location along each of the three axes.

5.7.3.3 Farsyte::Matrix::ThreeVec::ThreeVec (MatMe const & *p*)

Duplicate Constructor for ThreeVec.

Parameters

p	ThreeVec to duplicate. Initializes this position to contain a duplicate of the provided position.

Note

Can be called with any appropriately dimensioned matrix.

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

- matrix.h

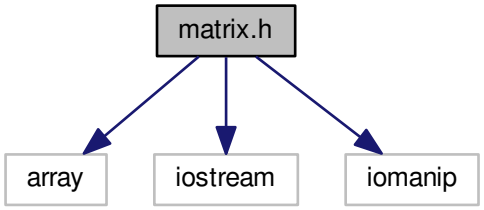
6 File Documentation

6.1 matrix.h File Reference

Matrix Library Exported API.

```
#include <array>
#include <iostream>
#include <iomanip>
```

Include dependency graph for matrix.h:



Classes

- class Farsyte::Matrix::Matrix< Nc, Nr, T >
  Matrix Template.
- class Farsyte::Matrix::ColVec< Nr, T >
  Column Vector Template.
- class Farsyte::Matrix::ThreeVec
  ThreeVec Class.

Functions

- template<int Nc, int Nr, typename T >
  bool **Farsyte::Matrix::operator==** (Matrix< Nc, Nr, T > const &L, Matrix< Nc, Nr, T > const &R)
  Equality Operator for Matrix-based Classes.
- template<int Nc, int Nr, typename T >
  bool **Farsyte::Matrix::operator!=** (Matrix< Nc, Nr, T > const &L, Matrix< Nc, Nr, T > const &R)
  Inequality Operator for Matrix-based Classes.
- template<int Nc, int Nr, typename T >
  Matrix< Nc, Nr, T > **Farsyte::Matrix::operator+** (Matrix< Nc, Nr, T > L, Matrix< Nc, Nr, T > const &R)
  Addition Operator for Matrix-based Classes.
- template<int Nc, int Nr, typename T >
  Matrix< Nc, Nr, T > **Farsyte::Matrix::operator-** (Matrix< Nc, Nr, T > L, Matrix< Nc, Nr, T > const &R)
  Addition Operator for Matrix-based Classes.
- template<int Nc, int Nr, typename T >
  Matrix< Nc, Nr, T > **Farsyte::Matrix::operator-** (Matrix< Nc, Nr, T > R)
  Negation Operator for Matrix-based Classes.
- template<int Nc, int Nr, typename T >
  Matrix< Nr, Nc, T > **Farsyte::Matrix::operator~** (Matrix< Nc, Nr, T > const &R)
  Transpose Operator for Matrix-based Classes.
- template<int Nc, int Ni, int Nr, typename T >
  Matrix< Nc, Nr, T > **Farsyte::Matrix::operator*** (Matrix< Ni, Nr, T > const &L, Matrix< Nc, Ni, T > const &R)
  Matrix Multiply.
- ThreeVec **Farsyte::Matrix::cross** (ThreeVec const &L, ThreeVec const &R)
  Cross-Product operation.

6.1.1 Detailed Description

Matrix Library Exported API.

This file provides data types, function prototypes and macro definitions for C code using this Matrix library.

Definition in file matrix.h.

6.2 matrix.h

```
00001 #ifndef _matrix_h
00002 #define _matrix_h
00003
00011 #include <array>
00012
00013 #include <iostream>
00014 using std::cerr;
00015 using std::endl;
00016
00017 #include <iomanip>
00018 using std::fixed;
00019 using std::setw;
00020 using std::setprecision;
00021
00022 namespace Farsyte
00023 {
00024   namespace Matrix
00025   {
00026
00036     template<int Nc, int Nr, typename T>
00037     class Matrix
00038     {
00039     public:
00040
00042       typedef T               value_type;
00043
00045       typedef T       &       reference;
00046
00048       typedef T const & const_reference;
00049
00051       typedef T       *       pointer;
00052
00054       typedef T const * const_pointer;
00055
00058       typedef std::array<T,Nr> C;
00059
00062       typedef std::array<C,Nc> A;
00063
00067       static size_t rows()
00068         {
00069           return Nr;
00070         }
00071
00075       static size_t cols()
00076         {
00077           return Nc;
00078         }
00079
00083       static size_t size()
00084         {
00085           return rows() * cols();
00086         }
00087
00088     protected:
00089
00096       T const & sub(int ri, int ci) const
00097         {
00098 #ifdef RANGE_CHECKER
00099           RANGE_CHECKER(1,ci,Nc);
00100           RANGE_CHECKER(1,ri,Nr);
00101 #endif
00102           return data[ci-1][ri-1];
00103         }
00104
00111       T       & sub(int ri, int ci)
00112         {
```

```
00113 #ifdef RANGE_CHECKER
00114           RANGE_CHECKER(1,ci,Nc);
00115           RANGE_CHECKER(1,ri,Nr);
00116 #endif
00117           return data[ci-1][ri-1];
00118         }
00119
00120     public:
00121
00126       Matrix()
00127         : data()
00128         {
00129         }
00130
00134       Matrix(T const &d)
00135         : data()
00136         {
00137           for (size_t i = 1; (i <= Nr) && (i <= Nc); ++i)
00138             sub(i,i) = d;
00139         }
00140
00150       Matrix(A const &a)
00151         : data(a)
00152         {
00153         }
00154
00160       Matrix(Matrix const &m)
00161         : data(m.data)
00162         {
00163         }
00164
00171       T const & operator()(int ri, int ci) const
00172         {
00173           return sub(ri, ci);
00174         }
00175
00182       T       & operator()(int ri, int ci)
00183         {
00184           return sub(ri, ci);
00185         }
00186
00191       bool equals(Matrix const &p) const
00192         {
00193           for (int ci = 1; ci <= Nc; ++ci)
00194             for (int ri = 1; ri <= Nr; ++ri)
00195               if (sub(ri,ci) != p.sub(ri,ci))
00196                 return false;
00197           return true;
00198         }
00199
00207       Matrix & increment_by(Matrix const &p)
00208         {
00209           for (int ci = 1; ci <= Nc; ++ci)
00210             for (int ri = 1; ri <= Nr; ++ri)
00211               sub(ri,ci) += p.sub(ri,ci);
00212           return *this;
00213         }
00214
00222       Matrix & operator+=(Matrix const &p)
00223         {
00224           return increment_by(p);
00225         }
00226
00234       Matrix & decrement_by(Matrix const &p)
00235         {
00236           for (int ci = 1; ci <= Nc; ++ci)
00237             for (int ri = 1; ri <= Nr; ++ri)
00238               sub(ri,ci) -= p.sub(ri,ci);
00239           return *this;
00240         }
00241
00249       Matrix & operator-=(Matrix const &p)
00250         {
00251           return decrement_by(p);
00252         }
00253
00257       Matrix negate()
00258         {
00259           for (int ci = 1; ci <= Nc; ++ci)
00260             for (int ri = 1; ri <= Nr; ++ri)
00261               sub(ri,ci) = -sub(ri,ci);
```

```
00262           return *this;
00263         }
00264
00268       Matrix<Nr,Nc,T>
00269       transpose() const
00270         {
00271           Matrix<Nr,Nc,T> R;
00272           for (int ci = 1; ci <= Nc; ++ci)
00273             for (int ri = 1; ri <= Nr; ++ri)
00274               R(ci,ri) = sub(ri,ci);
00275           return R;
00276         }
00277
00278     protected:
00280       A                         data;
00281     };
00282
00288     template<int Nc, int Nr, typename T>
00289     inline bool operator==(
00290       Matrix<Nc,Nr,T> const &L,
00291       Matrix<Nc,Nr,T> const &R)
00292     {
00293       return L.equals(R);
00294     }
00295
00301     template<int Nc, int Nr, typename T>
00302     inline bool operator!=(
00303       Matrix<Nc,Nr,T> const &L,
00304       Matrix<Nc,Nr,T> const &R)
00305     {
00306       return !(L == R);
00307     }
00308
00314     template<int Nc, int Nr, typename T>
00315     inline Matrix<Nc,Nr,T> operator+(
00316       Matrix<Nc,Nr,T> L,
00317       Matrix<Nc,Nr,T> const &R)
00318     {
00319       return L += R;
00320     }
00321
00327     template<int Nc, int Nr, typename T>
00328     inline Matrix<Nc,Nr,T> operator-(
00329       Matrix<Nc,Nr,T> L,
00330       Matrix<Nc,Nr,T> const &R)
00331     {
00332       return L -= R;
00333     }
00334
00339     template<int Nc, int Nr, typename T>
00340     inline Matrix<Nc,Nr,T> operator-(
00341       Matrix<Nc,Nr,T> R)
00342     {
00343       return R.negate();
00344     }
00345
00350     template<int Nc, int Nr, typename T>
00351     inline Matrix<Nr,Nc,T> operator~(
00352       Matrix<Nc,Nr,T> const &R)
00353     {
00354       return R.transpose();
00355     }
00356
00362     template<int Nc, int Ni, int Nr, typename T>
00363     inline Matrix<Nc,Nr,T>
00364     operator*(
00365       Matrix<Ni,Nr,T> const & L,
00366       Matrix<Nc,Ni,T> const & R)
00367     {
00368       Matrix<Nc,Nr,T> X;
00369       for (int ri=1; ri<=Nr; ++ri) {
00370         for (int ci=1; ci<=Nc; ++ci) {
00371           T &acc(X(ri,ci));
00372           acc = L(ri,1)*R(1,ci);
00373           for (int ii=2; ii<=Ni; ++ii) {
00374             acc += L(ri,ii)*R(ii,ci);
00375           }
00376         }
00377       }
00378       return X;
00379     }
```

```
00380
00389     template<int Nr, typename T>
00390     class ColVec
00391       : public Matrix<1,Nr,T>
00392     {
00393
00394     public:
00395
00397       typedef Matrix<1,Nr,T>            MatMe;
00398
00401       typedef typename MatMe::A A;
00402
00405       typedef typename MatMe::C C;
00406
00407     protected:
00408
00414       T const & sub(int ri) const
00415         {
00416           return MatMe::sub(ri, 1);
00417         }
00418
00424       T       & sub(int ri)
00425         {
00426           return MatMe::sub(ri, 1);
00427         }
00428
00429     public:
00430
00431       ColVec()
00432         : MatMe()
00433         {
00434         }
00435
00441       ColVec(A const &a)
00442         : MatMe(a)
00443         {
00444         }
00445
00451       ColVec(C const &a)
00452         : MatMe(A{{a}})
00453         {
00454         }
00455
00461       ColVec(MatMe const &c)
00462         : MatMe(c)
00463         {
00464         }
00465
00471       T const & operator()(int ri) const
00472         {
00473           return sub(ri);
00474         }
00475
00481       T       & operator()(int ri)
00482         {
00483           return sub(ri);
00484         }
00485
00486     };
00487
00495     class ThreeVec
00496       : public ColVec<3,double>
00497     {
00498     protected:
00500       typedef ColVec<3,double>  ColMe;
00501
00503       typedef ColMe::MatMe      MatMe;
00504
00507       typedef typename ColMe::C C;
00508
00509     public:
00510
00516       ThreeVec();
00517
00525       ThreeVec(double x, double y, double z);
00526
00532       ThreeVec(MatMe const &p);
00533
00534     };
00535
00541     extern ThreeVec cross(
```

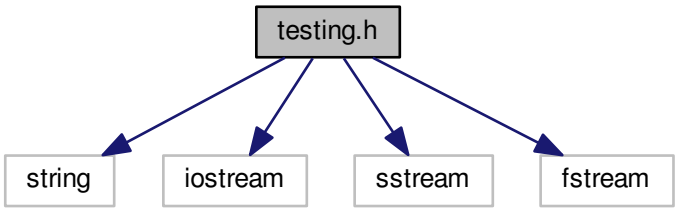
```
00542       ThreeVec const &L,
00543       ThreeVec const &R);
00544
00545   }
00546 }
00547
00548 #endif//_matrix_h
```

6.3 testing.h File Reference

Testing Library Exported API.

```
#include <string>
#include <iostream>
#include <sstream>
#include <fstream>
```

Include dependency graph for testing.h:



Classes

- class Farsyte::Testing::Log

  The Log Object.
- class Farsyte::Testing::Suite

  The Suite Object.
- class Farsyte::Testing::Test

  The Test object.
- class Farsyte::Testing::Oops

  The Oops Object.

Functions

- std::ostream & operator<< (std::ostream &s, Farsyte::Testing::Oops const &f)

  Dump information from an Oops onto an output stream.

6.3.1 Detailed Description

Testing Library Exported API.

This file provides data types, function prototypes and macro definitions for C code using this Testing library.

Definition in file testing.h.

6.4 testing.h

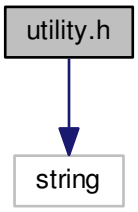
```
00001 #ifndef _testing_h
00002 #define _testing_h
00003
00011 #include <string>
00012 #include <iostream>
00013 #include <sstream>
00014 #include <fstream>
00015
00016 namespace Farsyte {
00017   namespace Testing {
00018
00019     class Log;
00020     class Suite;
00021     class Test;
00022     class Oops;
00023
00033     class Log {
00034     public:
00035
00037       std::ostream & out;
00038
00040       std::string const name;
00041
00043       Suite *curr;
00044
00046       size_t suites;
00047
00049       size_t tests;
00050
00052       size_t failed_tests;
00053
00055       size_t skipped_tests;
00056
00058       size_t errored_tests;
00059
00061       size_t total_fails;
00062
00064       size_t total_skips;
00065
00067       size_t total_errors;
00068
00074       Log(std::ostream &out, std::string const &name);
00075
00079       ~Log();
00080     };
00081
00092     class Suite {
00093     public:
00094
00096       Log &ref;
00097
00099       std::string const name;
00100
00102       Test *curr;
00103
00105       size_t tests;
00106
00108       size_t failed_tests;
00109
00111       size_t skipped_tests;
00112
00114       size_t errored_tests;
00115
00117       size_t total_fails;
00118
00120       size_t total_skips;
00121
00123       size_t total_errors;
00124
00133       Suite(Log &ref, std::string const &name);
00134
00141       ~Suite();
```

```
00142     };
00143
00163     class Test
00164       : public std::ostringstream
00165     {
00166     public:
00168       Suite &ref;
00169
00171       std::string const name;
00172
00174       size_t fails;
00175
00177       size_t skips;
00178
00180       size_t errors;
00181
00190       Test(Suite &ref, std::string const &name);
00191
00201       void fail(std::string const &cond);
00202
00212       void skip(std::string const &cond);
00213
00223       void error(std::string const &cond);
00224
00234       void pass(std::string const &cond);
00235
00246       ~Test();
00247     };
00248
00258     class Oops {
00259     public:
00260       std::string         file;
00261       int                 line;
00262       std::string         cond;
00263       mutable bool        pend;
00270       Oops(
00271         std::string     f,
00272         int             l,
00273         std::string     c);
00274
00283       virtual std::ostream& print(std::ostream&s) const;
00284
00291       virtual void cancel() const;
00292
00299       virtual ~Oops();
00300     };
00301   }
00302 }
00303
00306 extern std::ostream& operator<< (std::ostream&s,
      Farsyte::Testing::Oops const &f);
00307
00308 #endif//_testing_h
```

6.5 utility.h File Reference

Testing Library Exported API.

`#include <string>`
Include dependency graph for utility.h:



Functions

- std::string **Farsyte::Utility::literal** (char ch)

  Convert a character into its literal representation.
- std::string **Farsyte::Utility::literal** (std::string const &str)

  Convert a character into its literal representation.

6.5.1 Detailed Description

Testing Library Exported API.

This file provides data types, function prototypes and macro definitions for C code using this Testing library.

Definition in file utility.h.

6.6 utility.h

```
00001 #ifndef _utility_h
00002 #define _utility_h
00003
00011 #include <string>
00012
00013 namespace Farsyte {
00014   namespace Utility {
00015
00024     std::string literal(char ch);
00025
00034     std::string literal(std::string const &str);
00035
00036   }
00037 }
00038
00039 #endif//_utility_h
```