GPML

0.1

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 math::Matrix< N > Class Template Reference

Matrix generic class.

`#include <Matrix.hpp>`

Public Member Functions

- Matrix (uint size, N fill)
- Matrix (uint rows, uint cols, N fill)
- Matrix (const Matrix &m)
- N at (uint r, uint c) const
- void set (uint r, uint c, N val)
- std::pair< uint, uint > shape () const
- uint rows () const
- uint cols () const
- uint size () const
- Matrix & operator= (const Matrix &m)
- Matrix & operator+= (const Matrix &m)
- Matrix & operator-= (const Matrix &m)
- Matrix & operator*= (const N &scal)
- Matrix & operator*= (const Matrix &m)
- Matrix & operator/= (const N &scal)
- ~Matrix ()

3.1.1 Detailed Description

template<typename N>
class math::Matrix< N >

Matrix generic class.

Author

Daniel Nichols

Date

October 2018

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Matrix() [1/3]

```
template<typename N >
math::Matrix< N >::Matrix (
            uint size,
            N fill )
```

Square matrix constructor. Creates a size∗size matrix with every value set to fill.

Parameters

size	- size of the rows and cols of the matrix
fill	- default value for every entry

3.1.2.2 Matrix() [2/3]

```
template<typename N >
math::Matrix< N >::Matrix (
            uint rows,
            uint cols,
            N fill )
```

Matrix constructor. Creates a rows∗cols matrix with every value set to fill.

Parameters

rows	- number of rows
cols	- number of cols
fill	- default value for every entry

3.1.2.3 Matrix() [3/3]

```
template<typename N >
math::Matrix< N >::Matrix (
            const Matrix< N > & m )
```

Copy constructor. Copies matrix m into new matrix.

Parameters

m	- matrix to be copied

3.1.2.4 ~Matrix()

```
template<typename N >
math::Matrix< N >::~Matrix ( )
```

Destructor. Deletes the matrix internally

3.1.3 Member Function Documentation

3.1.3.1 at()

```
template<typename N >
N math::Matrix< N >::at (
            uint r,
            uint c ) const
```

Get element at r, c of the matrix 0-indexed.

Parameters

r	- row of return element
c	- column of return element

Exceptions

invalid_argument	thrown if r<0 or r>=rows() or c<0 or c>=cols()

3.1.3.2 cols()

```
template<typename N>
uint math::Matrix< N >::cols ( ) const  [inline]
```

Get the number of columns in the matrix.

Returns

the number of columns in the matrix

3.1.3.3 operator*=() [1/2]

```
template<typename N >
Matrix< N > & math::Matrix< N >::operator*= (
            const N & scal )
```

Adds Multiplies `this` by scaler `scal`

Parameters

scal	- scaler to multiply `this` by

Returns

a pointer to `m` after multiplication

3.1.3.4 operator*=() [2/2]

```
template<typename N >
Matrix< N > & math::Matrix< N >::operator*= (
            const Matrix< N > & m )
```

Performs matrix multiplication between `this` and `m`. This operation will throw an exception if `cols!=m.rows()`. It will also reshape `this` to that `rows()` does not change and `cols()` becomes `m.cols()`.

Parameters

m	- matrix to multiply by `this`.

Returns

a pointer to `this` after multiplication.

Exceptions

invalid_argument	if `cols()!=m.rows()` matrix multiplication is undefined

3.1.3.5 operator+=()

```
template<typename N >
Matrix< N > & math::Matrix< N >::operator+= (
            const Matrix< N > & m )
```

Adds matrix `m` to `this` element-wise

Parameters

m	- matrix to add to `this`. rows and cols must be equivalent.

Returns

a pointer to `this` after addition

Exceptions

invalid_argument	thrown if `rows()!=m.rows()` or `cols()!=m.cols()`

3.1.3.6 operator-=()

```
template<typename N >
Matrix< N > & math::Matrix< N >::operator-= (
            const Matrix< N > & m )
```

Subtracts matrix `m` from `this` element-wise

Parameters

m	- matrix to subtract from `this`. rows and cols must be equivalent.

Returns

a pointer to `this` after subtraction

Exceptions

invalid_argument	thrown if `rows()!=m.rows()` or `cols()!=m.cols()`

3.1.3.7 operator/=()

```
template<typename N>
Matrix& math::Matrix< N >::operator/= (
            const N & scal )
```

Divides `this` by scalar `scal` element-wise. Does not check for `scal==0` as division for class `N` might have non-standard definition.

Parameters

scal	- scalar to divide `this` by

Returns

a pointer to `m` after division

3.1.3.8 operator=()

```
template<typename N >
Matrix< N > & math::Matrix< N >::operator= (
            const Matrix< N > & m )
```

Copies `m` into `this`. Performs an element-wise copy. Ignores self-copy.

Parameters

m	- matrix to copy into `this`

Returns

pointer to `this` after copy

3.1.3.9 rows()

```
template<typename N>
uint math::Matrix< N >::rows ( ) const  [inline]
```

Get the number of rows in the matrix.

Returns

the number of rows in the matrix

3.1.3.10 set()

```
template<typename N >
void math::Matrix< N >::set (
            uint r,
            uint c,
            N val )
```

Set element at r, c of the matrix 0-indexed.

Parameters

r	- row of element set
c	- column of element set
val	- value to set element at r,c

Exceptions

invalid_argument	thrown if r<0 or r>=rows() or c<0 or c>=cols()

3.1.3.11 shape()

```
template<typename N>
std::pair<uint, uint> math::Matrix< N >::shape ( ) const  [inline]
```

Get the shape or (rows, cols). This is equivalent to `std::make_pair(rows(), cols());`

Returns

an STL pair containing the row count and column count

3.1.3.12 size()

```
template<typename N>
uint math::Matrix< N >::size ( ) const  [inline]
```

Get the size of the matrix (`size() == rows() * cols()`)

Returns

the size of the matrix

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

- /home/daniel/dev/cpp/math/include/Matrix.hpp

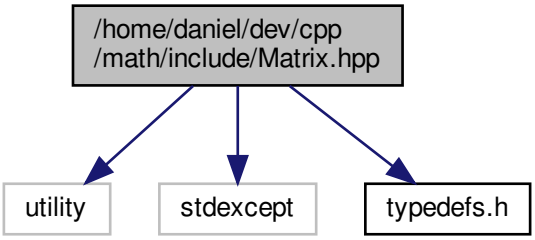
Chapter 4

File Documentation

4.1 /home/daniel/dev/cpp/math/include/Matrix.hpp File Reference

```
#include <utility>
#include <stdexcept>
#include "typedefs.h"
```

Include dependency graph for Matrix.hpp:



Classes

- class math::Matrix< N >

  Matrix generic class.

Typedefs

- typedef Matrix< int > math::iMatrix
- typedef Matrix< float > math::fMatrix
- typedef Matrix< double > math::dMatrix

Functions

- template<typename N >
  Matrix< N > math::operator+ (Matrix< N > lhs, const Matrix< N > &rhs)
- template<typename N >
  Matrix< N > math::operator- (Matrix< N > lhs, const Matrix< N > &rhs)
- template<typename N >
  Matrix< N > math::operator* (Matrix< N > lhs, const N &rhs)
- template<typename N >
  Matrix< N > math::operator* (const N &lhs, Matrix< N > rhs)
- template<typename N >
  Matrix< N > math::operator/ (Matrix< N > lhs, const N &rhs)
- template<typename N >
  Matrix< N > math::operator* (Matrix< N > lhs, const Matrix< N > &rhs)

4.1.1 Detailed Description

Contains Matrix class definition and implementation.

Author

Daniel Nichols

Date

October 2018

4.1.2 Typedef Documentation

4.1.2.1 dMatrix

```
typedef Matrix<double> math::dMatrix
```

double precision matrix

4.1.2.2 fMatrix

```
typedef Matrix<float> math::fMatrix
```

float precision matrix

4.1.2.3 iMatrix

```
typedef Matrix<int> math::iMatrix
```

integer matrix

4.1.3 Function Documentation

4.1.3.1 operator∗() [1/3]

```
template<typename N >
Matrix< N > math::operator* (
            Matrix< N > lhs,
            const N & rhs )
```

Multiplies lhs and scalar rhs. Copies lhs and multiplies by scalar rhs

Parameters

lhs	- left hand side matrix
rhs	- right hand side scalar

Returns

a new matrix with elements multiplication of `lhs` and `rhs`

4.1.3.2 operator∗() [2/3]

```
template<typename N >
Matrix< N > math::operator* (
            const N & lhs,
            Matrix< N > rhs )
```

Multiplies scalar lhs and matrix rhs. Copies rhs and multiplies by scalar lhs

Parameters

lhs	- left hand side scalar
rhs	- right hand side matrix

Returns

a new matrix with elements multiplication of `lhs` and `rhs`

4.1.3.3 operator∗() [3/3]

```
template<typename N >
Matrix<N> math::operator* (
```

```
            Matrix< N > lhs,
            const Matrix< N > & rhs )
```

Performs matrix multiplication of `rhs` and `lhs`

Parameters

lhs	- left hand side matrix
rhs	- right hand side matrix

Returns

a new matrix resulting from matrix multiplication. Result will have shape `lhs.rows(), rhs.cols()`.

Exceptions

invalid_argument	if `lhs.cols() != rhs.rows()`

4.1.3.4 operator+()

```
template<typename N >
Matrix< N > math::operator+ (
            Matrix< N > lhs,
            const Matrix< N > & rhs )
```

Adds lhs and rhs matrices element-wise. Copies lhs and add rhs to it.

Parameters

lhs	- left hand side matrix of addition
rhs	- right hand side matrix of addition

Returns

a new matrix with elements from element-wise addition of `lhs` and `rhs`

Exceptions

invalid_argument	if `lhs` and `rhs` do not have same shape

4.1.3.5 operator-()

```
template<typename N >
Matrix< N > math::operator- (
```

```
            Matrix< N > lhs,
            const Matrix< N > & rhs )
```

Subtracts lhs and rhs matrices element-wise. Copies lhs and subtract rhs from it.

Parameters

lhs	- left hand side matrix of subtraction
rhs	- right hand side matrix of subtraction

Returns

a new matrix with elements from element-wise subtraction of `lhs` and `rhs`

Exceptions

invalid_argument	if `lhs` and `rhs` do not have same shape

4.1.3.6 operator/()

```
template<typename N >
Matrix<N> math::operator/ (
            Matrix< N > lhs,
            const N & rhs )
```

Divides lhs matrix by scalar rhs. Copies lhs and divides by scalar rhs. Does not check if rhs is zero due to unknown type of `N`.

Parameters

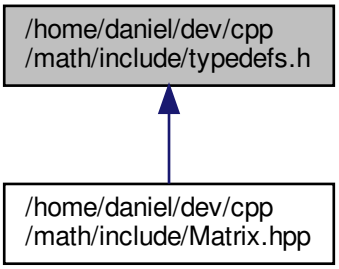
lhs	- left hand side matrix
rhs	- right hand side scalar

Returns

a new matrix with elements division of `lhs` and `rhs`

4.2 /home/daniel/dev/cpp/math/include/typedefs.h File Reference

This graph shows which files directly or indirectly include this file:



Typedefs

- typedef unsigned int math::uint

4.2.1 Detailed Description

Defines utilility types for math library.

Author

Daniel Nichols

Date

October 2018

4.2.2 Typedef Documentation

4.2.2.1 uint

```
typedef unsigned int math::uint
```

shorthand for unsigned int type

Index