My Project

Generated by Doxygen 1.12.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 matrix Namespace Reference

Classes

- class Matrix

  A class for performing operations on 2D matrices such as addition, subtraction, and multiplication.
- class vector

  A class for performing vector operations such as addition, subtraction, scaling, dot product, magnitude, and cosine similarity.

Chapter 5

Class Documentation

5.1 matrix::Matrix Class Reference

A class for performing operations on 2D matrices such as addition, subtraction, and multiplication.

`#include <vector.h>`

Static Public Member Functions

- static std::vector< std::vector< int > > add2D (const std::vector< std::vector< int > > &firstMatrix, const std::vector< std::vector< int > > &secondMatrix)

  Adds two 2D matrices element-wise.
- static std::vector< std::vector< int > > substract2D (const std::vector< std::vector< int > > &firstMatrix, const std::vector< std::vector< int > > &secondMatrix)

  Subtracts the second 2D matrix from the first 2D matrix element-wise.
- static std::vector< std::vector< int > > multiply2D (const std::vector< std::vector< int > > &firstMatrix, const std::vector< std::vector< int > > &secondMatrix)

  Multiplies two 2D matrices.

5.1.1 Detailed Description

A class for performing operations on 2D matrices such as addition, subtraction, and multiplication.

Definition at line 138 of file vector.h.

5.1.2 Member Function Documentation

5.1.2.1 add2D()

```
static std::vector< std::vector< int > > matrix::Matrix::add2D (
            const std::vector< std::vector< int > > & firstMatrix,
            const std::vector< std::vector< int > > & secondMatrix)  [inline], [static]
```

Adds two 2D matrices element-wise.

Parameters

firstMatrix	The first matrix.
secondMatrix	The second matrix.

Returns

The resultant matrix after addition.

Exceptions

std::invalid_argument	if the matrices have different dimensions.

Definition at line 147 of file vector.h.

5.1.2.2 multiply2D()

```
static std::vector< std::vector< int > > matrix::Matrix::multiply2D (
            const std::vector< std::vector< int > > & firstMatrix,
            const std::vector< std::vector< int > > & secondMatrix)  [inline], [static]
```

Multiplies two 2D matrices.

Parameters

firstMatrix	The first matrix.
secondMatrix	The second matrix.

Returns

The resultant matrix after multiplication.

Exceptions

std::invalid_argument	if the number of columns in the first matrix is not equal to the number of rows in the second matrix.

Definition at line 205 of file vector.h.

5.1.2.3 substract2D()

```
static std::vector< std::vector< int > > matrix::Matrix::substract2D (
            const std::vector< std::vector< int > > & firstMatrix,
            const std::vector< std::vector< int > > & secondMatrix)  [inline], [static]
```

Subtracts the second 2D matrix from the first 2D matrix element-wise.

Parameters

firstMatrix	The first matrix.
secondMatrix	The second matrix.

Returns

The resultant matrix after subtraction.

Exceptions

std::invalid_argument	if the matrices have different dimensions.

Definition at line 176 of file vector.h.

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

- vector.h

5.2 matrix::vector Class Reference

A class for performing vector operations such as addition, subtraction, scaling, dot product, magnitude, and cosine similarity.

`#include <vector.h>`

Static Public Member Functions

- static std::vector< int > add (std::vector< int > firstVector, std::vector< int > secondVector)
  Adds two vectors element-wise.
- static std::vector< int > substract (std::vector< int > firstVector, std::vector< int > secondVector)
  Subtracts the second vector from the first vector element-wise.
- static std::vector< int > scale (std::vector< int > vector, int scale)
  Scales a vector by a given factor.
- static int dotProduct (const std::vector< int > &firstVector, const std::vector< int > &secondVector=std::vector< int >())
  Computes the dot product of two vectors.
- static double magnitude (const std::vector< int > &firstVector, const std::vector< int > &secondVector=std::vector< int >())
  Computes the magnitude of the difference between two vectors.
- static double cosineSimilarity (const std::vector< int > &firstVector, const std::vector< int > &secondVector)
  Computes the cosine similarity between two vectors.

5.2.1 Detailed Description

A class for performing vector operations such as addition, subtraction, scaling, dot product, magnitude, and cosine similarity.

Definition at line 15 of file vector.h.

5.2.2 Member Function Documentation

5.2.2.1 add()

```
static std::vector< int > matrix::vector::add (
            std::vector< int > firstVector,
            std::vector< int > secondVector)  [inline], [static]
```

Adds two vectors element-wise.

Parameters

firstVector	The first vector.
secondVector	The second vector.

Returns

The resultant vector after addition.

Definition at line 23 of file vector.h.

5.2.2.2 cosineSimilarity()

```
static double matrix::vector::cosineSimilarity (
            const std::vector< int > & firstVector,
            const std::vector< int > & secondVector)  [inline], [static]
```

Computes the cosine similarity between two vectors.

Parameters

firstVector	The first vector.
secondVector	The second vector.

Returns

The cosine similarity between the two vectors.

Exceptions

std::invalid_argument	if the vectors are of different lengths.

Definition at line 122 of file vector.h.

5.2.2.3 dotProduct()

```
static int matrix::vector::dotProduct (
            const std::vector< int > & firstVector,
            const std::vector< int > & secondVector = std::vector<int>())  [inline], [static]
```

Computes the dot product of two vectors.

Parameters

firstVector	The first vector.
secondVector	The second vector. Defaults to an empty vector (treated as a zero vector).

Returns

The dot product of the two vectors.

Exceptions

std::invalid_argument	if the vectors are of different lengths.

Definition at line 73 of file vector.h.

5.2.2.4 magnitude()

```
static double matrix::vector::magnitude (
            const std::vector< int > & firstVector,
            const std::vector< int > & secondVector = std::vector<int>())  [inline], [static]
```

Computes the magnitude of the difference between two vectors.

Parameters

firstVector	The first vector.
secondVector	The second vector. Defaults to an empty vector (treated as a zero vector).

Returns

The magnitude of the difference between the two vectors.

Exceptions

std::invalid_argument	if the vectors are of different lengths.

Definition at line 99 of file vector.h.

5.2.2.5 scale()

```
static std::vector< int > matrix::vector::scale (
            std::vector< int > vector,
            int scale)  [inline], [static]
```

Scales a vector by a given factor.

Parameters

vector	The vector to scale.
scale	The scaling factor.

Returns

The resultant scaled vector.

Definition at line 59 of file vector.h.

5.2.2.6 substract()

```
static std::vector< int > matrix::vector::substract (
            std::vector< int > firstVector,
            std::vector< int > secondVector)  [inline], [static]
```

Subtracts the second vector from the first vector element-wise.

Parameters

firstVector	The first vector.
secondVector	The second vector.

Returns

The resultant vector after subtraction.

Definition at line 41 of file vector.h.

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

- vector.h

Chapter 6

File Documentation

6.1 vector.cpp File Reference

```
#include "vector.h"
```

Functions

- int main ()

  The main function demonstrating various vector and matrix operations.

6.1.1 Function Documentation

6.1.1.1 main()

```
int main ()
```

The main function demonstrating various vector and matrix operations.

This function performs the following operations:

- Addition and subtraction of two vectors.
- Addition, subtraction, and multiplication of 2D matrices.
- Calculation of the dot product of a vector.
- Calculation of the cosine similarity between two vectors.

Returns

int Returns 0 on successful execution.

Definition at line 44 of file vector.cpp.

6.2 vector.cpp

Go to the documentation of this file.

```
00001
00031 #include "vector.h"
00032
00044 int main() {
00045     // Vectors for addition and subtraction
00046     std::vector<int> firstVector = {1, 3, 6, 9};
00047     std::vector<int> secondVector = {1, 3, -6, 9};
00048
00049     // Perform vector addition
00050     std::vector<int> resultant = matrix::vector::add(firstVector, secondVector);
00051
00052     // Perform vector subtraction
00053     std::vector<int> resultantt = matrix::vector::substract(firstVector, secondVector);
00054
00055     // 2D matrices for addition and subtraction
00056     std::vector<std::vector<int» firstMatrix = {{1, 3, 6, 9},
00057                                                 {5, 9, 8, 0},
00058                                                 {1, 9, 7, 5}};
00059     std::vector<std::vector<int» secondMatrix = {{1, 3, 6, 9},
00060                                                  {5, 9, 8, 0},
00061                                                  {1, 9, 7, 5}};
00062
00063     // Perform 2D matrix addition
00064     matrix::Matrix::add2D(firstMatrix, secondMatrix);
00065
00066     // Perform 2D matrix subtraction
00067     matrix::Matrix::substract2D(firstMatrix, secondMatrix);
00068
00069     // 2D matrices for multiplication
00070     std::vector<std::vector<int» firstMatrixForMultiplication = {{1, 3, 6, 9},
00071                                                                  {5, 9, 8, 0},
00072                                                                  {1, 9, 7, 5}};
00073     std::vector<std::vector<int» secondMatrixForMultiplication = {{1, 3,  6, 9},
00074                                                                   {5, 9,  8, 0},
00075                                                                   {1, 9,  7, 5},
00076                                                                   {1, 18, 9, 8}};
00077
00078     // Perform 2D matrix multiplication
00079     matrix::Matrix::multiply2D(firstMatrixForMultiplication, secondMatrixForMultiplication);
00080
00081     // Vector for dot product calculation
00082     std::vector<int> vec1 = {1, 2, 3};
00083
00084     // Calculate dot product of a vector with a zero vector (default)
00085     int result2 = matrix::vector::dotProduct(vec1);
00086
00087     // Calculate cosine similarity between two vectors
00088     std::cout « matrix::vector::cosineSimilarity(firstVector, secondVector) * 100 « " %" « std::endl;
00089
00090     return 0;
00091 }
```

6.3 vector.h File Reference

```
#include <utility>
#include <vector>
#include <iostream>
#include <cmath>
```

Classes

- class matrix::vector

  A class for performing vector operations such as addition, subtraction, scaling, dot product, magnitude, and cosine similarity.

- class matrix::Matrix

  A class for performing operations on 2D matrices such as addition, subtraction, and multiplication.

Namespaces

- namespace matrix

6.4 vector.h

Go to the documentation of this file.

```
00001 #ifndef VECTOR_VECTOR_H
00002 #define VECTOR_VECTOR_H
00003
00004 #include <utility>
00005 #include <vector>
00006 #include <iostream>
00007 #include <cmath>
00008
00009 namespace matrix {
00010
00015     class vector {
00016     public:
00023         static std::vector<int> add(std::vector<int> firstVector, std::vector<int> secondVector) {
00024             std::vector<int> resultant(firstVector.size());
00025             for (int i = 0; i < firstVector.size(); i++) {
00026                 resultant[i] = firstVector[i] + secondVector[i];
00027             }
00028             for (int i = 0; i < firstVector.size(); i++) {
00029                 std::cout « resultant[i] « " ";
00030             }
00031             std::cout « std::endl;
00032             return resultant;
00033         }
00034
00041         static std::vector<int> substract(std::vector<int> firstVector, std::vector<int> secondVector)
    {
00042             std::vector<int> resultant(firstVector.size());
00043             for (int i = 0; i < firstVector.size(); i++) {
00044                 resultant[i] = firstVector[i] - secondVector[i];
00045             }
00046             for (int i = 0; i < firstVector.size(); i++) {
00047                 std::cout « resultant[i] « " ";
00048             }
00049             std::cout « std::endl;
00050             return resultant;
00051         }
00052
00059         static std::vector<int> scale(std::vector<int> vector, int scale) {
00060             for (int i = 0; i < vector.size(); i++) {
00061                 vector[i] *= scale;
00062             }
00063             return vector;
00064         }
00065
00073         static int dotProduct(const std::vector<int> &firstVector, const std::vector<int>
    &secondVector = std::vector<int>()) {
00074             std::vector<int> adjustedSecondVector = secondVector;
00075
00076             if (adjustedSecondVector.empty()) {
00077                 adjustedSecondVector.resize(firstVector.size(), 0);
00078             }
00079
00080             if (firstVector.size() != adjustedSecondVector.size()) {
00081                 throw std::invalid_argument("Vectors must be of same length");
00082             }
00083
00084             int result = 0;
00085             for (size_t i = 0; i < firstVector.size(); ++i) {
00086                 result += firstVector[i] * adjustedSecondVector[i];
00087             }
00088
00089             return result;
00090         }
00091
00099         static double magnitude(const std::vector<int> &firstVector, const std::vector<int>
    &secondVector = std::vector<int>()) {
00100             std::vector<int> adjustedSecondVector = secondVector;
00101
00102             if (adjustedSecondVector.empty()) {
00103                 adjustedSecondVector.resize(firstVector.size(), 0);
00104             }
00105             if (firstVector.size() != adjustedSecondVector.size()) {
00106                 throw std::invalid_argument("Vectors must be of same length");
```

```
00107             }
00108             int result = 0;
00109             for (int i = 0; i < firstVector.size(); i++) {
00110                 result += (adjustedSecondVector[i] - firstVector[i]) * (adjustedSecondVector[i] -
      firstVector[i]);
00111             }
00112             return sqrt((double) result);
00113         }
00114
00122         static double cosineSimilarity(const std::vector<int> &firstVector, const std::vector<int>
      &secondVector) {
00123             if (firstVector.size() != secondVector.size()) {
00124                 throw std::invalid_argument("Vectors must be of same length");
00125             }
00126             int dotProduct = matrix::vector::dotProduct(firstVector, secondVector);
00127             double magnitudeOfA = matrix::vector::magnitude(firstVector);
00128             double magnitudeOfB = matrix::vector::magnitude(secondVector);
00129             double cosineSimilarity = dotProduct / (magnitudeOfA * magnitudeOfB);
00130             return cosineSimilarity;
00131         }
00132     };
00133
00138     class Matrix {
00139     public:
00147         static std::vector<std::vector<int> add2D(const std::vector<std::vector<int> &firstMatrix,
      const std::vector<std::vector<int> &secondMatrix) {
00148             if (firstMatrix.size() != secondMatrix.size() || firstMatrix[0].size() !=
      secondMatrix[0].size()) {
00149                 throw std::invalid_argument("Matrices must have the same dimensions");
00150             }
00151
00152             std::vector<std::vector<int> resultant(firstMatrix.size(),
      std::vector<int>(firstMatrix[0].size()));
00153
00154             for (size_t i = 0; i < firstMatrix.size(); ++i) {
00155                 for (size_t j = 0; j < firstMatrix[i].size(); ++j) {
00156                     resultant[i][j] = firstMatrix[i][j] + secondMatrix[i][j];
00157                 }
00158             }
00159             for (const auto &row: resultant) {
00160                 for (const int element: row) {
00161                     std::cout « element « "   ";
00162                 }
00163                 std::cout « std::endl;
00164             }
00165
00166             return resultant;
00167         }
00168
00176         static std::vector<std::vector<int> substract2D(const std::vector<std::vector<int>
      &firstMatrix, const std::vector<std::vector<int> &secondMatrix) {
00177             if (firstMatrix.size() != secondMatrix.size() || firstMatrix[0].size() !=
      secondMatrix[0].size()) {
00178                 throw std::invalid_argument("Matrices must have the same dimensions");
00179             }
00180
00181             std::vector<std::vector<int> resultant(firstMatrix.size(),
      std::vector<int>(firstMatrix[0].size()));
00182
00183             for (size_t i = 0; i < firstMatrix.size(); i++) {
00184                 for (size_t j = 0; j < firstMatrix[i].size(); j++) {
00185                     resultant[i][j] = firstMatrix[i][j] - secondMatrix[i][j];
00186                 }
00187             }
00188             for (const auto &row: resultant) {
00189                 for (const int element: row) {
00190                     std::cout « element « "   ";
00191                 }
00192                 std::cout « std::endl;
00193             }
00194
00195             return resultant;
00196         }
00197
00205         static std::vector<std::vector<int> multiply2D(const std::vector<std::vector<int>
      &firstMatrix, const std::vector<std::vector<int> &secondMatrix) {
00206             if (firstMatrix[0].size() != secondMatrix.size()) {
00207                 throw std::invalid_argument("The number of columns in the first matrix must be equal
      to the number of rows in the second matrix");
00208             }
00209             std::vector<std::vector<int> resultant(firstMatrix.size(),
      std::vector<int>(secondMatrix[0].size()));
00210             for (size_t i = 0; i < firstMatrix.size(); i++) {
00211                 for (size_t j = 0; j < firstMatrix[i].size(); j++) {
00212                     resultant[i][j] = 0;
00213                     for (size_t k = 0; k < secondMatrix.size(); k++) {
00214                         resultant[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
```

```
00215                     }
00216                 }
00217             }
00218
00219             for (const auto &row: resultant) {
00220                 for (const int element: row) {
00221                     std::cout « element « "   ";
00222                 }
00223                 std::cout « std::endl;
00224             }
00225
00226             return resultant;
00227         }
00228     };
00229 }
00230
00231 #endif //VECTOR_VECTOR_H
```

Index