

My Project

Generated by Doxygen 1.8.16

1 Class Index	1
1.1 Class List	1
2 Class Documentation	3
2.1 Vector Class Reference	3
2.1.1 Detailed Description	3
2.1.2 Constructor & Destructor Documentation	3
2.1.2.1 Vector() [1/2]	4
2.1.2.2 Vector() [2/2]	4
2.1.3 Member Function Documentation	4
2.1.3.1 add()	4
2.1.3.2 angle()	5
2.1.3.3 cross()	5
2.1.3.4 dot()	6
2.1.3.5 equal()	6
2.1.3.6 getI()	7
2.1.3.7 getJ()	7
2.1.3.8 getK()	7
2.1.3.9 norm()	7
2.1.3.10 output()	7
2.1.3.11 setI()	8
2.1.3.12 setJ()	8
2.1.3.13 setK()	9
2.1.3.14 sub()	9
Index	11

Chapter 1

Class Index

1.1 Class List

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

Vector	3
----------------------------------	---

Chapter 2

Class Documentation

2.1 Vector Class Reference

```
#include <Vector.hpp>
```

Public Member Functions

- [Vector](#) ()
- [Vector](#) (double vx, double vy, double vz)
- double [getI](#) () const
- double [getJ](#) () const
- double [getK](#) () const
- void [setI](#) (double newVx)
- void [setJ](#) (double newVy)
- void [setK](#) (double newVz)
- bool [equal](#) (const [Vector](#) &rhs) const
- [Vector](#) [add](#) (const [Vector](#) &rhs) const
- [Vector](#) [sub](#) (const [Vector](#) &rhs) const
- [Vector](#) [cross](#) (const [Vector](#) &rhs) const
- double [dot](#) (const [Vector](#) &rhs) const
- double [norm](#) () const
- double [angle](#) (const [Vector](#) &rhs) const
- void [output](#) (std::ostream &out) const

2.1.1 Detailed Description

This is a basic C++ class to represent three-dimensional numbers. It's not meant to be difficult but as a refresher on classes.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 `Vector()` [1/2]

```
Vector::Vector ( )
```

Default constructor. It should set the scalar components to 0.

2.1.2.2 `Vector()` [2/2]

```
Vector::Vector (
    double vx,
    double vy,
    double vz )
```

And a second one. Use the parameters to set the scalar components.

Parameters

<code>vx</code>	- the scalar value to use for i component
<code>vy</code>	- the scalar value to use for j component
<code>vz</code>	- the scalar value to use for k component

2.1.3 Member Function Documentation

2.1.3.1 `add()`

```
Vector Vector::add (
    const Vector & rhs ) const
```

Creates and returns a new [Vector](#) object representing the vector addition of two [Vector](#) objects

Returns

a new [Vector](#) object that contains the appropriate summed components

Parameters

<i>rhs</i>	- the Vector object to add to this object.
------------	--

2.1.3.2 angle()

```
double Vector::angle (  
    const Vector & rhs ) const
```

Returns the angle between two [Vector](#) objects in radians (over interval $[0, 2\pi)$).

Parameters

<i>rhs</i>	- the Vector object to find the angle between with this object.
------------	---

Returns

the angle (-1 if angle undefined)

2.1.3.3 cross()

```
Vector Vector::cross (  
    const Vector & rhs ) const
```

Creates and returns a new [Vector](#) object that is cross product of this and the given [Vector](#) object.

Returns

a new [Vector](#) object that contains the cross product of this and the given [Vector](#) object.

Parameters

<i>rhs</i>	- the Vector object to cross with this object.
------------	--

2.1.3.4 dot()

```
double Vector::dot (
    const Vector & rhs ) const
```

Creates and returns a new [Vector](#) object that is dot product of this and the given [Vector](#) object.

Returns

the dot product of this and the given [Vector](#) object.

Parameters

<i>rhs</i>	- the Vector object to dot with this object.
------------	--

2.1.3.5 equal()

```
bool Vector::equal (
    const Vector & rhs ) const
```

Returns true if the scalar components for this object and rhs are the same, false otherwise.

Returns

true if scalar components in both objects are the same.

2.1.3.6 getI()

```
double Vector::getI ( ) const
```

Returns the scalar of the i component

Returns

vx.

2.1.3.7 getJ()

```
double Vector::getJ ( ) const
```

Returns the scalar of the j component

Returns

vy.

2.1.3.8 getK()

```
double Vector::getK ( ) const
```

Returns the scalar of the k component

Returns

vz.

2.1.3.9 norm()

```
double Vector::norm ( ) const
```

Returns the norm of the [Vector](#) object.

Returns

the norm (-1 if magnitude undefined)

2.1.3.10 output()

```
void Vector::output (
    std::ostream & out ) const
```

Outputs this [Vector](#) object on the given ostream. `"vxi + vyj + vzk"` (for debugging).

Parameters

<i>out</i>	- the os-tream object to use to output.
------------	---

2.1.3.11 setI()

```
void Vector::setI (
    double newVx )
```

Updates the scalar of the i component to the given newVx parameter.

Parameters

<i>newVx</i>	- the new value to use for the vx field.
--------------	--

2.1.3.12 setJ()

```
void Vector::setJ (
    double newVy )
```

Updates the scalar of the i component to the given newVx parameter.

Parameters

<i>newVy</i>	- the new value to use for the vx field.
--------------	--

2.1.3.13 setK()

```
void Vector::setK (
    double newVz )
```

Updates the scalar of the i component to the given newVx parameter.

Parameters

<i>newVz</i>	- the new value to use for the vx field.
--------------	--

2.1.3.14 sub()

```
Vector Vector::sub (
    const Vector & rhs ) const
```

Creates and returns a new [Vector](#) object representing the vector subtraction of two [Vector](#) objects

Returns

a new [Vector](#) object that contains the appropriate difference components

Parameters

<i>rhs</i>	- the Vector object to subtract from this object.
------------	---

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

- Vector.hpp

Index

- add
 - Vector, [4](#)
- angle
 - Vector, [5](#)
- cross
 - Vector, [5](#)
- dot
 - Vector, [6](#)
- equal
 - Vector, [6](#)
- getI
 - Vector, [6](#)
- getJ
 - Vector, [7](#)
- getK
 - Vector, [7](#)
- norm
 - Vector, [7](#)
- output
 - Vector, [7](#)
- setI
 - Vector, [8](#)
- setJ
 - Vector, [8](#)
- setK
 - Vector, [8](#)
- sub
 - Vector, [9](#)
- Vector, [3](#)
 - add, [4](#)
 - angle, [5](#)
 - cross, [5](#)
 - dot, [6](#)
 - equal, [6](#)
 - getI, [6](#)
 - getJ, [7](#)
 - getK, [7](#)
 - norm, [7](#)
 - output, [7](#)
 - setI, [8](#)
 - setJ, [8](#)
 - setK, [8](#)
 - sub, [9](#)
 - Vector, [3](#), [4](#)