Ejemplo clases POO

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

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 documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Complex Class Reference

Represents a complex number and provides operations for complex arithmetic.

`#include <Complex.h>`

Public Member Functions

- Complex (void)
  Default constructor for the Complex class.
- Complex (double re, double im=0.0)
  Constructor with real and imaginary parts.
- Complex (const Complex &other)
  Copy constructor for the Complex class.
- float add (double a, double b)
  Adds two double values.
- int add (int a, int b)
  Adds two integer values.
- void SetData (void)
  Sets the data for the complex number.
- void SetReal (double re)
  Sets the real part of the complex number.
- void SetImag (double im)
  Sets the imaginary part of the complex number.
- double GetReal (void)
  Gets the real part of the complex number.
- double GetImag (void)
  Gets the imaginary part of the complex number.
- Complex operator+ (const Complex &other)
  Overloads the addition operator for complex numbers.
- Complex operator+ ()
  Unary plus operator overload.
- Complex operator- (const Complex &other)
  Overloads the subtraction operator for complex numbers.

- Complex operator∗ (const Complex &other)

  Overloads the multiplication operator for complex numbers.
- Complex operator/ (const Complex &other)

  Overloads the division operator for complex numbers.
- Complex & operator= (const Complex &other)

  Overloads the assignment operator for complex numbers.
- void display ()

  Displays the details of the complex number.

Public Attributes

- string nombre

  Name associated with the complex number.

Friends

- int operator== (const Complex &lhs, const Complex &rhs)

  Overloads the equality operator for complex numbers.
- int operator!= (const Complex &lhs, const Complex &rhs)

  Overloads the inequality operator for complex numbers.
- ostream & operator<< (ostream &os, const Complex &c)

  Overloads the insertion operator for output streams.

4.1.1 Detailed Description

Represents a complex number and provides operations for complex arithmetic.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Complex() [1/2]

```
Complex::Complex (
            double re,
            double im = 0.0 )
```

Constructor with real and imaginary parts.

Parameters

re	Real part of the complex number.
im	Imaginary part of the complex number (default is 0.0).

4.1.2.2 Complex() [2/2]

```
Complex::Complex (
            const Complex & other )
```

Copy constructor for the Complex class.

Parameters

other	Another Complex object to copy from.

4.1.3 Member Function Documentation

4.1.3.1 add() [1/2]

```
float Complex::add (
            double a,
            double b )
```

Adds two double values.

Parameters

a	First value.
b	Second value.

Returns

Sum of the two values as a float.

4.1.3.2 add() [2/2]

```
int Complex::add (
            int a,
            int b )
```

Adds two integer values.

Parameters

a	First value.
b	Second value.

Returns

Sum of the two values as an integer.

4.1.3.3 GetImag()

```
double Complex::GetImag (
            void  )  [inline]
```

Gets the imaginary part of the complex number.

Returns

Imaginary part of the complex number.

4.1.3.4 GetReal()

```
double Complex::GetReal (
            void  )  [inline]
```

Gets the real part of the complex number.

Returns

Real part of the complex number.

4.1.3.5 operator*()

```
Complex Complex::operator* (
            const Complex & other )
```

Overloads the multiplication operator for complex numbers.

Parameters

other	Another Complex object to multiply.

Returns

Product of the two complex numbers.

4.1.3.6 operator+() [1/2]

```
Complex Complex::operator+ ( )
```

Unary plus operator overload.

Returns

The same Complex object.

4.1.3.7 operator+() [2/2]

```
Complex Complex::operator+ (
            const Complex & other )
```

Overloads the addition operator for complex numbers.

Parameters

other	Another Complex object to add.

Returns

Sum of the two complex numbers.

4.1.3.8 operator-()

```
Complex Complex::operator- (
            const Complex & other )
```

Overloads the subtraction operator for complex numbers.

Parameters

other	Another Complex object to subtract.

Returns

Difference of the two complex numbers.

4.1.3.9 operator/()

```
Complex Complex::operator/ (
            const Complex & other )
```

Overloads the division operator for complex numbers.

Parameters

other	Another Complex object to divide by.

Returns

Quotient of the two complex numbers.

4.1.3.10 operator=()

```
Complex & Complex::operator= (
            const Complex & other )
```

Overloads the assignment operator for complex numbers.

Parameters

other	Another Complex object to assign from.

Returns

Reference to the assigned Complex object.

4.1.3.11 SetImag()

```
void Complex::SetImag (
            double im )
```

Sets the imaginary part of the complex number.

Parameters

im	Imaginary part to set.

4.1.3.12 SetReal()

```
void Complex::SetReal (
            double re )
```

Sets the real part of the complex number.

Parameters

re	Real part to set.

4.1.4 Friends And Related Function Documentation

4.1.4.1 operator"!=

```
int operator!= (
            const Complex & lhs,
            const Complex & rhs )  [friend]
```

Overloads the inequality operator for complex numbers.

Parameters

lhs	Left-hand side Complex object.
rhs	Right-hand side Complex object.

Returns

1 if the two complex numbers are not equal, 0 otherwise.

4.1.4.2 operator<<

```
ostream& operator<< (
            ostream & os,
            const Complex & c )  [friend]
```

Overloads the insertion operator for output streams.

Parameters

os	Output stream.
c	Complex object to insert into the stream.

Returns

Reference to the output stream.

4.1.4.3 operator==

```
int operator== (
            const Complex & lhs,
            const Complex & rhs )  [friend]
```

Overloads the equality operator for complex numbers.

Parameters

lhs	Left-hand side Complex object.
rhs	Right-hand side Complex object.

Returns

1 if the two complex numbers are equal, 0 otherwise.

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

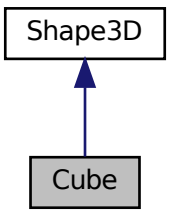
- include/Complex.h
- src/Complex.cpp

4.2 Cube Class Reference

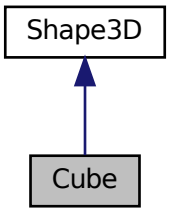
Representa un cubo en 3D, derivado de Shape3D.

```
#include <Cube.h>
```

Inheritance diagram for Cube:



Collaboration diagram for Cube:



Public Member Functions

- Cube (double len)

  Constructor que inicializa el cubo con una longitud dada.
- Cube ()

  Constructor por defecto. Inicializa el cubo con longitud cero.
- double getVolume () const override

  Calcula el volumen del cubo.
- double getSurfaceArea () const override

  Calcula el área superficial del cubo.
- void setLength (double len)

  Establece la longitud del lado del cubo.
- double getLength () const

  Obtiene la longitud del lado del cubo.
- ~Cube () override

  Destructor de la clase Cube.

4.2.1 Detailed Description

Representa un cubo en 3D, derivado de Shape3D.

Proporciona métodos para calcular el volumen y el área superficial, así como para manipular la longitud de sus lados.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Cube()

```
Cube::Cube (
            double len )
```

Constructor que inicializa el cubo con una longitud dada.

Parameters

len	Longitud del lado del cubo.

4.2.3 Member Function Documentation

4.2.3.1 getLength()

```
double Cube::getLength ( ) const
```

Obtiene la longitud del lado del cubo.

Returns

Longitud del lado.

4.2.3.2 getSurfaceArea()

```
double Cube::getSurfaceArea ( ) const  [override], [virtual]
```

Calcula el área superficial del cubo.

Returns

Área superficial del cubo.

Implements Shape3D.

4.2.3.3 getVolume()

```
double Cube::getVolume ( ) const  [override], [virtual]
```

Calcula el volumen del cubo.

Returns

Volumen del cubo.

Implements Shape3D.

4.2.3.4 setLength()

```
void Cube::setLength (
            double len )
```

Establece la longitud del lado del cubo.

Parameters

len	Nueva longitud del lado.

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

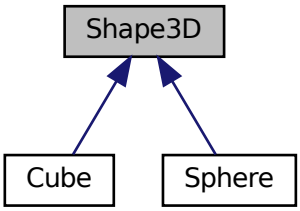
- include/Cube.h
- src/Cube.cpp

4.3 Shape3D Class Reference

Interface for three-dimensional geometric shapes.

`#include <Shape3d.h>`

Inheritance diagram for Shape3D:



Public Member Functions

- virtual double getVolume () const =0

  Compute the volume of the shape.
- virtual double getSurfaceArea () const =0

  Compute the surface area of the shape.
- virtual ~Shape3D ()

  Virtual destructor.

4.3.1 Detailed Description

Interface for three-dimensional geometric shapes.

Shape3D is an abstract base class that requires derived classes to implement methods for computing the volume and the surface area of a shape. It defines a virtual destructor to ensure proper cleanup of derived objects through base pointers.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ~Shape3D()

```
virtual Shape3D::~Shape3D ( )  [inline], [virtual]
```

Virtual destructor.

Ensures that destructors of derived classes are called when an object is deleted through a pointer to Shape3D.

4.3.3 Member Function Documentation

4.3.3.1 getSurfaceArea()

```
virtual double Shape3D::getSurfaceArea ( ) const  [pure virtual]
```

Compute the surface area of the shape.

Pure virtual method that should be implemented by concrete shape classes. The returned value represents the total surface area.

Returns

Surface area as a double.

Implemented in Sphere, and Cube.

4.3.3.2 getVolume()

```
virtual double Shape3D::getVolume ( ) const  [pure virtual]
```

Compute the volume of the shape.

This is a pure virtual function. Derived classes must override this method and return the volume in the appropriate units.

Returns

Volume as a double.

Implemented in Sphere, and Cube.

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

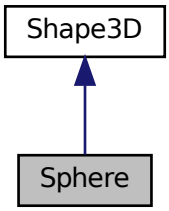
- include/Shape3d.h

4.4 Sphere Class Reference

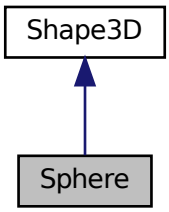
Concrete implementation of a three-dimensional sphere.

`#include <Sphere.h>`

Inheritance diagram for Sphere:



Collaboration diagram for Sphere:



Public Member Functions

- Sphere (double len)
  Construct a Sphere with a specified radius.
- Sphere ()
  Default constructor.
- double getVolume () const override
  Compute the volume of the sphere.
- double getSurfaceArea () const override
  Compute the surface area of the sphere.
- void setLength (double len)
  Set the sphere radius.
- double getLength () const
  Get the sphere radius.
- ~Sphere () override
  Virtual destructor.

4.4.1 Detailed Description

Concrete implementation of a three-dimensional sphere.

Sphere represents a geometric sphere defined by its radius. It implements the Shape3D interface by providing concrete implementations for computing volume and surface area.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Sphere() [1/2]

```
Sphere::Sphere (
            double len )
```

Construct a Sphere with a specified radius.

If the provided radius is not positive, the implementation will typically fall back to a default positive radius (see implementation in Sphere.cpp).

Parameters

len	Radius of the sphere (must be $>$ 0 to be accepted).

4.4.2.2 Sphere() [2/2]

```
Sphere::Sphere ( )
```

Default constructor.

Constructs a sphere with a default radius (usually 1.0).

4.4.2.3 ~Sphere()

```
Sphere::~Sphere ( )  [override]
```

Virtual destructor.

Ensures proper cleanup of resources. Marked override to match the Shape3D virtual destructor.

4.4.3 Member Function Documentation

4.4.3.1 getLength()

```
double Sphere::getLength ( ) const
```

Get the sphere radius.

Returns the current radius stored in the object.

Returns

Radius as a double.

4.4.3.2 getSurfaceArea()

```
double Sphere::getSurfaceArea ( ) const  [override], [virtual]
```

Compute the surface area of the sphere.

Implements Shape3D::getSurfaceArea(). Returns the surface area using the formula: 4 ∗ pi ∗ r^2.

Returns

Surface area as a double.

Implements Shape3D.

4.4.3.3 getVolume()

```
double Sphere::getVolume ( ) const  [override], [virtual]
```

Compute the volume of the sphere.

Implements Shape3D::getVolume(). Returns the volume using the formula: (4/3) ∗ pi ∗ r^3.

Returns

Volume as a double.

Implements Shape3D.

4.4.3.4 setLength()

```
void Sphere::setLength (
            double len )
```

Set the sphere radius.

Sets the radius to the provided value if it is positive. If the value is non-positive, the radius is left unchanged.

Parameters

len	New radius value (must be > 0 to take effect).

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

- include/Sphere.h
- src/Sphere.cpp

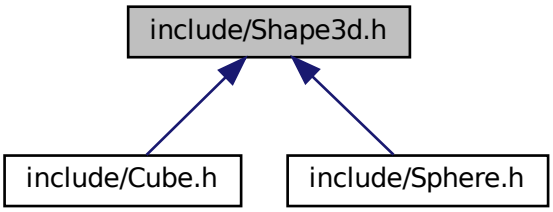
Chapter 5

File Documentation

5.1 include/Shape3d.h File Reference

Abstract base class for 3D shapes.

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



Classes

- class Shape3D

  Interface for three-dimensional geometric shapes.

5.1.1 Detailed Description

Abstract base class for 3D shapes.

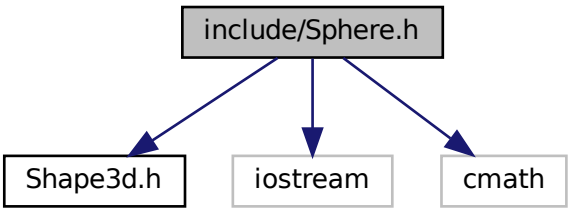
This header defines the Shape3D interface which provides two pure virtual methods that all concrete 3D shape classes must implement: volume and surface area calculations.

5.2 include/Sphere.h File Reference

Declaration of the Sphere concrete 3D shape.

```
#include "Shape3d.h"
#include <iostream>
#include <cmath>
```

Include dependency graph for Sphere.h:



Classes

- class Sphere

  Concrete implementation of a three-dimensional sphere.

5.2.1 Detailed Description

Declaration of the Sphere concrete 3D shape.

The Sphere class implements the Shape3D interface and provides methods to compute the volume and surface area of a sphere. The class stores the sphere size in the private member `radious` (the radius of the sphere).

Index