My Project

Generated by Doxygen 1.8.17

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

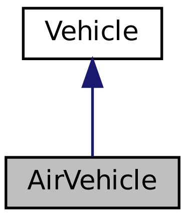
Class Documentation

3.1 AirVehicle Class Reference

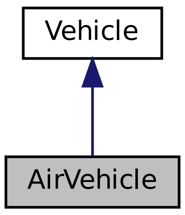
Represents an air vehicle, derived from the Vehicle class.

```
#include <AirVehicle.h>
```

Inheritance diagram for AirVehicle:



Collaboration diagram for AirVehicle:



Public Member Functions

- AirVehicle (string b, int s, int alt)

  Constructor for the AirVehicle class.
- void display () const override

  Displays the details of the air vehicle.

Additional Inherited Members

3.1.1 Detailed Description

Represents an air vehicle, derived from the Vehicle class.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 AirVehicle()

```
AirVehicle::AirVehicle (
              string b,
              int s,
              int alt )
```

Constructor for the AirVehicle class.

Parameters

b	Brand of the vehicle.
s	Speed of the vehicle.
alt	Altitude of the air vehicle.

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

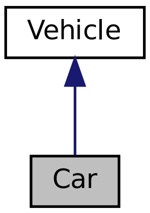
- include/AirVehicle.h
- src/AirVehicle.cpp

3.2 Car Class Reference

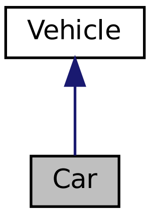
Represents a car, derived from the Vehicle class.

```
#include <Car.h>
```

Inheritance diagram for Car:



Collaboration diagram for Car:



Public Member Functions

- Car (string b, int s, int doors, int wheels=4)

    Constructor for the Car class.
- ∼Car () override

    Destructor for the Car class.
- int getNumDoors () const

    Gets the number of doors in the car.
- void setNumDoors (int doors)

    Sets the number of doors in the car.
- int getNumWheels () const

    Gets the number of wheels in the car.
- void setNumWheels (int wheels)

    Sets the number of wheels in the car.
- void display () const override

    Displays the details of the car.

Additional Inherited Members

3.2.1 Detailed Description

Represents a car, derived from the Vehicle class.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Car()

```
Car::Car (
            string b,
            int s,
            int doors,
            int wheels = 4 )
```

Constructor for the Car class.

Parameters

b	Brand of the car.
s	Speed of the car.
doors	Number of doors in the car.
wheels	Number of wheels in the car (default is 4).

3.2.3 Member Function Documentation

3.2.3.1 getNumDoors()

```
int Car::getNumDoors ( ) const
```

Gets the number of doors in the car.

Returns

Number of doors.

3.2.3.2 getNumWheels()

```
int Car::getNumWheels ( ) const
```

Gets the number of wheels in the car.

Returns

Number of wheels.

3.2.3.3 setNumDoors()

```
void Car::setNumDoors (
             int doors )
```

Sets the number of doors in the car.

Parameters

doors	Number of doors to set.

3.2.3.4 setNumWheels()

```
void Car::setNumWheels (
             int wheels )
```

Sets the number of wheels in the car.

Parameters

wheels	Number of wheels to set.

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

- include/Car.h
- src/Car.cpp

3.3 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.

3.3.1 Detailed Description

Represents a complex number and provides operations for complex arithmetic.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 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).

3.3.2.2 Complex() `[2/2]`

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

Copy constructor for the Complex class.

Parameters

other	Another Complex object to copy from.

3.3.3 Member Function Documentation

3.3.3.1 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.

3.3.3.2 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.

3.3.3.3 GetImag()

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

Gets the imaginary part of the complex number.

Returns

Imaginary part of the complex number.

3.3.3.4 GetReal()

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

Gets the real part of the complex number.

Returns

Real part of the complex number.

3.3.3.5 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.

3.3.3.6 operator+() [1/2]

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

Unary plus operator overload.

Returns

The same Complex object.

3.3.3.7 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.

3.3.3.8 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.

3.3.3.9 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.

3.3.3.10 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.

3.3.3.11 SetImag()

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

Sets the imaginary part of the complex number.

Parameters

im	Imaginary part to set.

3.3.3.12 SetReal()

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

Sets the real part of the complex number.

Parameters

re	Real part to set.

3.3.4 Friends And Related Function Documentation

3.3.4.1 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.

3.3.4.2 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.

3.3.4.3 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

3.4 Cube Class Reference

Represents a cube with methods to calculate its volume and surface area.

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

Public Member Functions

- Cube (double l=1.0)

    Constructor for the Cube class.
- double getLength () const

    Gets the length of the cube's side.
- void setLength (double l)

    Sets the length of the cube's side.
- double getVolume () const

    Calculates the volume of the cube.
- double getSurfaceArea () const

    Calculates the surface area of the cube.

Public Attributes

- string colour

    Colour of the cube.

3.4.1 Detailed Description

Represents a cube with methods to calculate its volume and surface area.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Cube()

```
Cube::Cube (
              double l = 1.0 )
```

Constructor for the Cube class.

Parameters

l	Length of the cube's side (default is 1.0).

3.4.3 Member Function Documentation

3.4.3.1 getLength()

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

Gets the length of the cube's side.

Returns

Length of the cube's side.

3.4.3.2 getSurfaceArea()

```
double Cube::getSurfaceArea ( ) const
```

Calculates the surface area of the cube.

Returns

Surface area of the cube.

3.4.3.3 getVolume()

```
double Cube::getVolume ( ) const
```

Calculates the volume of the cube.

Returns

Volume of the cube.

3.4.3.4 setLength()

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

Sets the length of the cube's side.

Parameters

l	Length to set.

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

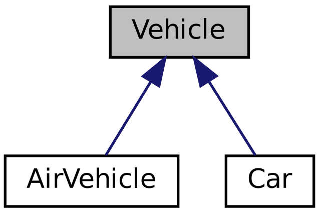
- include/Cube.h
- src/Cube.cpp

3.5 Vehicle Class Reference

Represents a generic vehicle with basic attributes and methods.

```
#include <Vehicle.h>
```

Inheritance diagram for Vehicle:



Public Member Functions

- Vehicle (string b, int s)

    Constructor for the Vehicle class.
- Vehicle (string b)

    Constructor for the Vehicle class with only the brand.
- virtual ∼Vehicle ()

    Virtual destructor for the Vehicle class.
- int getSpeed () const

    Gets the speed of the vehicle.
- void setSpeed (int s)

    Sets the speed of the vehicle.
- virtual void display () const

    Displays the details of the vehicle.

Protected Attributes

- string brand

    Brand of the vehicle.
- int speed

    Speed of the vehicle in km/h.

3.5.1 Detailed Description

Represents a generic vehicle with basic attributes and methods.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Vehicle() `[1/2]`

```
Vehicle::Vehicle (
            string b,
            int s )
```

Constructor for the Vehicle class.

Parameters

b	Brand of the vehicle.
s	Speed of the vehicle in km/h.

3.5.2.2 Vehicle() `[2/2]`

```
Vehicle::Vehicle (
            string b )
```

Constructor for the Vehicle class with only the brand.

Parameters

b	Brand of the vehicle.

3.5.2.3 ∼Vehicle()

```
Vehicle::∼Vehicle ( )  [virtual]
```

Virtual destructor for the Vehicle class.

Note

This is important for proper cleanup in polymorphic use cases.

3.5.3 Member Function Documentation

3.5.3.1 display()

```
void Vehicle::display ( ) const  [virtual]
```

Displays the details of the vehicle.

Note

This is a virtual method and can be overridden by derived classes.

Reimplemented in Car, and AirVehicle.

3.5.3.2 getSpeed()

```
int Vehicle::getSpeed ( ) const
```

Gets the speed of the vehicle.

Returns

Speed of the vehicle in km/h.

3.5.3.3 setSpeed()

```
void Vehicle::setSpeed (
             int s )
```

Sets the speed of the vehicle.

Parameters

s	Speed to set in km/h.

3.5.4 Member Data Documentation

3.5.4.1 brand

```
string Vehicle::brand  [protected]
```

Brand of the vehicle.

Note

This attribute is protected, so it is accessible by derived classes.

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

- include/Vehicle.h
- src/Vehicle.cpp

Index