

Course project in Programming Languages

1.0

Generated by Doxygen 1.9.1

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 Airport Class Reference	7
4.1.1 Detailed Description	9
4.1.2 Member Function Documentation	9
4.1.2.1 get_airport()	9
4.1.2.2 get_city()	9
4.1.2.3 get_fuel_cost_per_liter()	9
4.1.2.4 get_lane_lengths()	10
4.1.2.5 get_location()	10
4.1.2.6 get_scheduled_flights()	10
4.1.3 Friends And Related Function Documentation	11
4.1.3.1 AirSpaceManager	11
4.1.3.2 operator<<	11
4.2 AirSpaceManager Class Reference	11
4.2.1 Detailed Description	12
4.2.2 Constructor & Destructor Documentation	12
4.2.2.1 AirSpaceManager()	12
4.2.2.2 ~AirSpaceManager()	13
4.2.3 Member Function Documentation	13
4.2.3.1 input_airport()	13
4.2.3.2 input_flight()	13
4.2.3.3 input_plane()	13
4.2.3.4 run()	13
4.3 DateTime Class Reference	14
4.3.1 Detailed Description	14
4.3.2 Constructor & Destructor Documentation	14
4.3.2.1 DateTime() [1/2]	14
4.3.2.2 DateTime() [2/2]	15
4.3.3 Friends And Related Function Documentation	15
4.3.3.1 AirSpaceManager	15
4.3.3.2 operator<<	15
4.4 ExportFile Class Reference	16
4.4.1 Constructor & Destructor Documentation	16
4.4.1.1 ExportFile()	16

4.4.1.2 ~ExportFile()	16
4.5 Flight Class Reference	17
4.5.1 Detailed Description	19
4.5.2 Member Function Documentation	19
4.5.2.1 assign_plane()	19
4.5.2.2 exhaust_rate() [1/2]	19
4.5.2.3 exhaust_rate() [2/2]	20
4.5.2.4 flight_duration() [1/2]	20
4.5.2.5 flight_duration() [2/2]	21
4.5.2.6 set_from()	21
4.5.2.7 set_to()	22
4.5.3 Friends And Related Function Documentation	22
4.5.3.1 AirSpaceManager	22
4.5.3.2 operator<<	23
4.6 IEnumerable Class Reference	23
4.6.1 Detailed Description	24
4.6.2 Constructor & Destructor Documentation	24
4.6.2.1 IEnumerable()	24
4.6.3 Member Function Documentation	24
4.6.3.1 get_id()	25
4.6.4 Friends And Related Function Documentation	25
4.6.4.1 AirSpaceManager	25
4.7 ImportFile Class Reference	25
4.7.1 Constructor & Destructor Documentation	26
4.7.1.1 ImportFile()	26
4.7.1.2 ~ImportFile()	26
4.8 Plane Class Reference	26
4.8.1 Detailed Description	27
4.8.2 Member Function Documentation	27
4.8.2.1 can_land_on()	28
4.8.2.2 get_average_speed()	29
4.8.2.3 max_flight_distance()	29
4.8.3 Friends And Related Function Documentation	29
4.8.3.1 AirSpaceManager	29
4.8.3.2 operator<<	29
4.9 Time Class Reference	30
4.9.1 Detailed Description	31
4.9.2 Constructor & Destructor Documentation	31
4.9.2.1 Time() [1/2]	31
4.9.2.2 Time() [2/2]	31
4.9.3 Member Function Documentation	32
4.9.3.1 from()	32

4.9.4 Friends And Related Function Documentation	32
4.9.4.1 operator<<	32
5 File Documentation	35
5.1 airport.cpp File Reference	35
5.1.1 Function Documentation	35
5.1.1.1 operator<<()	35
5.2 airport.hpp File Reference	36
5.2.1 Detailed Description	37
5.3 airspace_manager.cpp File Reference	37
5.4 airspace_manager.hpp File Reference	37
5.4.1 Detailed Description	38
5.5 date_time.cpp File Reference	38
5.5.1 Function Documentation	39
5.5.1.1 operator<<() [1/2]	39
5.5.1.2 operator<<() [2/2]	40
5.6 date_time.hpp File Reference	40
5.6.1 Detailed Description	41
5.7 enumerable.cpp File Reference	42
5.8 enumerable.hpp File Reference	42
5.8.1 Detailed Description	43
5.9 export_file.cpp File Reference	43
5.10 export_file.hpp File Reference	44
5.10.1 Detailed Description	44
5.11 flight.cpp File Reference	45
5.11.1 Function Documentation	45
5.11.1.1 operator<<()	45
5.12 flight.hpp File Reference	45
5.12.1 Detailed Description	46
5.13 import_file.cpp File Reference	47
5.14 import_file.hpp File Reference	47
5.14.1 Detailed Description	48
5.15 main.cpp File Reference	48
5.15.1 Function Documentation	49
5.15.1.1 main()	49
5.16 plane.cpp File Reference	49
5.16.1 Function Documentation	50
5.16.1.1 operator<<()	50
5.17 plane.hpp File Reference	50
5.17.1 Detailed Description	51
5.18 project_fwd_def.hpp File Reference	51
5.18.1 Detailed Description	51

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

AirSpaceManager	11
DateTime	14
ExportFile	16
IEnumerable	23
Airport	7
Flight	17
Plane	26
ImportFile	25
Time	30

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Airport	Represents an airport with details such as city, name, location, lane lengths, fuel cost, and scheduled flights	7
AirSpaceManager	A class to manage airports, planes, and flights for optimizing airline resources	11
DateTime	Represents a specific point in time with detailed components like year, month, day, hour, minute, and second	14
ExportFile	16
Flight	Represents a flight, including its schedule, route, and assigned plane	17
IEnumerable	A base class that provides unique IDs for objects	23
ImportFile	25
Plane	Represents an aircraft with details for efficient airline management	26
Time	Represents a time duration or specific time of day without date information	30

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

airport.cpp	35
airport.hpp	
Contains the declaration of the Airport class and its methods	36
airspace_manager.cpp	37
airspace_manager.hpp	
Contains the declaration of the AirSpaceManager class and its methods for managing airports, planes, and flights	37
date_time.cpp	38
date_time.hpp	
Contains the declaration of the DateTime and Time classes and their methods for handling date and time information	40
enumerable.cpp	42
enumerable.hpp	
Contains the declaration of the IEnumerable class, which provides a unique identifier for derived objects	42
export_file.cpp	43
export_file.hpp	
Contains a wrapper for export file	44
flight.cpp	45
flight.hpp	
Contains the declaration of the Flight class, representing a flight with associated data and operations	45
import_file.cpp	47
import_file.hpp	
Contains a wrapper for import file	47
main.cpp	48
plane.cpp	49
plane.hpp	
Contains the declaration of the Plane class and its methods	50
project_fwd_def.hpp	
Contains the declaration of the project_fwd_def class and its methods	51

Chapter 4

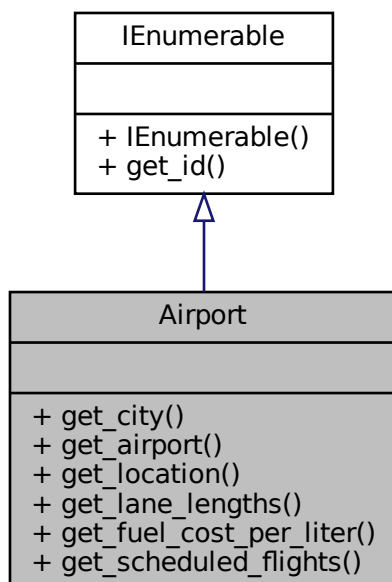
Data Structure Documentation

4.1 Airport Class Reference

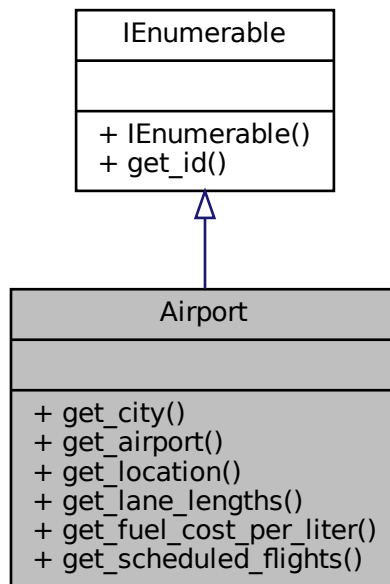
Represents an airport with details such as city, name, location, lane lengths, fuel cost, and scheduled flights.

```
#include <airport.hpp>
```

Inheritance diagram for Airport:



Collaboration diagram for Airport:



Public Member Functions

- `const std::string & get_city () const`
Gets the city where the airport is located.
- `const std::string & get_airport () const`
Gets the name of the airport.
- `const std::string & get_location () const`
Gets the geographical location of the airport.
- `const std::vector< double > & get_lane_lengths () const`
Gets the lengths of the lanes at the airport.
- `double get_fuel_cost_per_liter () const`
Gets the cost of fuel per liter at the airport.
- `const std::vector< Flight * > & get_scheduled_flights () const`
Gets the list of scheduled flights at the airport.

Friends

- class [AirSpaceManager](#)
Allows [AirSpaceManager](#) to access private members of the [Airport](#) class.
- `std::ostream & operator<< (std::ostream &out, const Airport &airport)`
Outputs the details of the [Airport](#) object to an output stream.

4.1.1 Detailed Description

Represents an airport with details such as city, name, location, lane lengths, fuel cost, and scheduled flights.

This class provides methods to access airport-related information and handles its associated data.

4.1.2 Member Function Documentation

4.1.2.1 get_airport()

```
const std::string & Airport::get_airport ( ) const
```

Gets the name of the airport.

Returns

A constant reference to the airport name.

4.1.2.2 get_city()

```
const std::string & Airport::get_city ( ) const
```

Gets the city where the airport is located.

Returns

A constant reference to the city name.

4.1.2.3 get_fuel_cost_per_liter()

```
double Airport::get_fuel_cost_per_liter ( ) const
```

Gets the cost of fuel per liter at the airport.

Returns

The fuel cost per liter.

4.1.2.4 get_lane_lengths()

```
const std::vector< double > & Airport::get_lane_lengths ( ) const
```

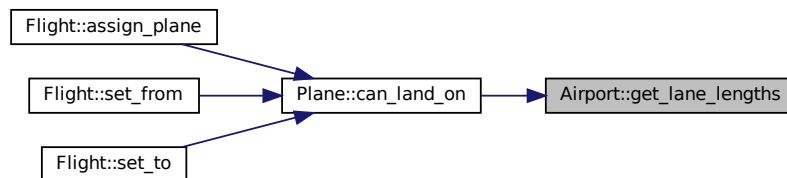
Gets the lengths of the lanes at the airport.

Returns

A constant reference to a vector containing the lane lengths.

Referenced by [Plane::can_land_on\(\)](#).

Here is the caller graph for this function:



4.1.2.5 get_location()

```
const std::string & Airport::get_location ( ) const
```

Gets the geographical location of the airport.

Returns

A constant reference to the location.

4.1.2.6 get_scheduled_flights()

```
const std::vector< Flight * > & Airport::get_scheduled_flights ( ) const
```

Gets the list of scheduled flights at the airport.

Returns

A constant reference to a vector of pointers to [Flight](#) objects.

4.1.3 Friends And Related Function Documentation

4.1.3.1 AirSpaceManager

```
friend class AirSpaceManager [friend]
```

Allows [AirSpaceManager](#) to access private members of the [Airport](#) class.

4.1.3.2 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Airport & airport ) [friend]
```

Outputs the details of the [Airport](#) object to an output stream.

Parameters

<i>out</i>	The output stream to write to.
<i>airport</i>	The Airport object to output.

Returns

A reference to the output stream.

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

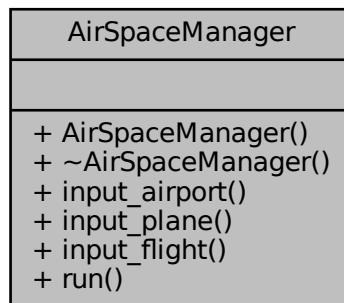
- [airport.hpp](#)
- [airport.cpp](#)

4.2 AirSpaceManager Class Reference

A class to manage airports, planes, and flights for optimizing airline resources.

```
#include <airspace_manager.hpp>
```

Collaboration diagram for AirSpaceManager:



Public Member Functions

- [AirSpaceManager](#) ()
Default constructor for [AirSpaceManager](#).
- [~AirSpaceManager](#) ()
Destructor for [AirSpaceManager](#). Frees allocated memory.
- void [input_airport](#) ()
Prompts the user to input details for a new airport.
- void [input_plane](#) ()
Prompts the user to input details for a new plane.
- void [input_flight](#) ()
Prompts the user to input details for a new flight.
- void [run](#) ()
Main entry point to run the [AirSpaceManager](#) application.

4.2.1 Detailed Description

A class to manage airports, planes, and flights for optimizing airline resources.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AirSpaceManager()

```
AirSpaceManager::AirSpaceManager ( )
```

Default constructor for [AirSpaceManager](#).

4.2.2.2 ~AirSpaceManager()

```
AirSpaceManager::~~AirSpaceManager ( )
```

Destructor for [AirSpaceManager](#). Frees allocated memory.

4.2.3 Member Function Documentation

4.2.3.1 input_airport()

```
void AirSpaceManager::input_airport ( )
```

Prompts the user to input details for a new airport.

4.2.3.2 input_flight()

```
void AirSpaceManager::input_flight ( )
```

Prompts the user to input details for a new flight.

4.2.3.3 input_plane()

```
void AirSpaceManager::input_plane ( )
```

Prompts the user to input details for a new plane.

4.2.3.4 run()

```
void AirSpaceManager::run ( )
```

Main entry point to run the [AirSpaceManager](#) application.

Referenced by [main\(\)](#).

Here is the caller graph for this function:



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

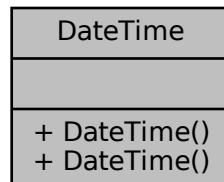
- [airspace_manager.hpp](#)
- [airspace_manager.cpp](#)

4.3 DateTime Class Reference

Represents a specific point in time with detailed components like year, month, day, hour, minute, and second.

```
#include <date_time.hpp>
```

Collaboration diagram for DateTime:



Public Member Functions

- [DateTime](#) ()
Default constructor initializing to an undefined date and time.
- [DateTime](#) (unsigned short year, unsigned short month, unsigned short day, unsigned short hour, unsigned short minute, unsigned short second)
Constructs a [DateTime](#) object with the specified date and time.

Friends

- class [AirSpaceManager](#)
- `std::ostream & operator<< (std::ostream &out, const DateTime &time)`
Overloaded output stream operator for [DateTime](#).

4.3.1 Detailed Description

Represents a specific point in time with detailed components like year, month, day, hour, minute, and second.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 DateTime() [1/2]

```
DateTime::DateTime ( )
```

Default constructor initializing to an undefined date and time.

4.3.2.2 DateTime() [2/2]

```
DateTime::DateTime (
    unsigned short year,
    unsigned short month,
    unsigned short day,
    unsigned short hour,
    unsigned short minute,
    unsigned short second )
```

Constructs a [DateTime](#) object with the specified date and time.

Parameters

<i>year</i>	The year of the date.
<i>month</i>	The month of the date (1-12).
<i>day</i>	The day of the date (1-31).
<i>hour</i>	The hour of the time (0-23).
<i>minute</i>	The minute of the time (0-59).
<i>second</i>	The second of the time (0-59).

4.3.3 Friends And Related Function Documentation

4.3.3.1 AirSpaceManager

```
friend class AirSpaceManager [friend]
```

4.3.3.2 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const DateTime & time ) [friend]
```

Overloaded output stream operator for [DateTime](#).

Parameters

<i>out</i>	The output stream.
<i>time</i>	The DateTime object to be output.

Returns

Reference to the output stream.

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

- [date_time.hpp](#)
- [date_time.cpp](#)

4.4 ExportFile Class Reference

```
#include <export_file.hpp>
```

Collaboration diagram for ExportFile:



Public Member Functions

- [ExportFile](#) (const std::string &)
- [~ExportFile](#) ()

4.4.1 Constructor & Destructor Documentation

4.4.1.1 ExportFile()

```
ExportFile::ExportFile (
    const std::string & filename )
```

4.4.1.2 ~ExportFile()

```
ExportFile::~~ExportFile ( )
```

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

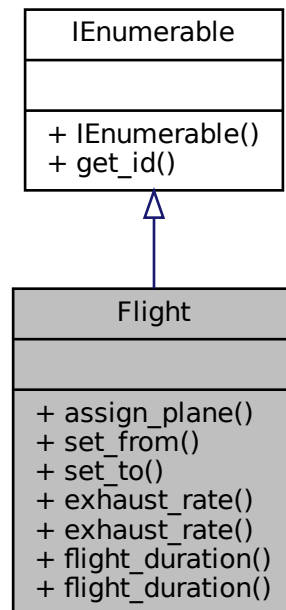
- [export_file.hpp](#)
- [export_file.cpp](#)

4.5 Flight Class Reference

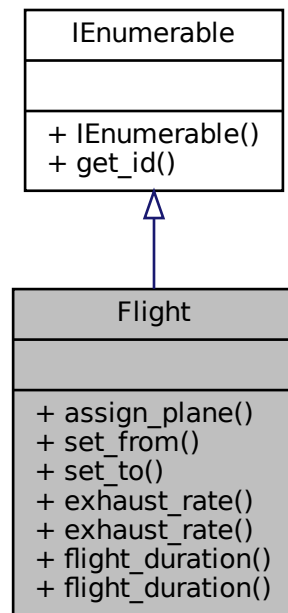
Represents a flight, including its schedule, route, and assigned plane.

```
#include <flight.hpp>
```

Inheritance diagram for Flight:



Collaboration diagram for Flight:



Public Member Functions

- bool `assign_plane` (const `Plane` &candidate)
Assigns a plane to the flight if it meets the necessary criteria.
- bool `set_from` (const `Airport` &from)
Sets the origin airport for the flight.
- bool `set_to` (const `Airport` &to)
Sets the destination airport for the flight.
- double `exhaust_rate` () const
Calculates the workload for the pilot(s) during this flight.
- double `exhaust_rate` (const `Plane` * _plane) const
Calculates the workload for a given plane during this flight.
- double `flight_duration` () const
Calculates the flight duration based on the route length and plane speed.
- double `flight_duration` (const `Plane` *) const
Calculates the flight duration based on the route length and plane speed.

Friends

- class `AirSpaceManager`
Declares `AirSpaceManager` as a friend class to access private members.
- `std::ostream & operator<<` (std::ostream &out, const `Flight` &flight)
Overloaded output stream operator for `Flight`.

4.5.1 Detailed Description

Represents a flight, including its schedule, route, and assigned plane.

The [Flight](#) class encapsulates details about a specific flight, such as the takeoff time, route length, origin and destination airports, and the plane assigned to the flight. It also provides methods to calculate workload and flight duration.

4.5.2 Member Function Documentation

4.5.2.1 `assign_plane()`

```
bool Flight::assign_plane (
    const Plane & candidate )
```

Assigns a plane to the flight if it meets the necessary criteria.

Parameters

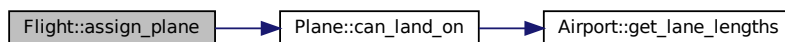
<i>candidate</i>	The plane to be considered for assignment.
------------------	--

Returns

True if the plane was successfully assigned, false otherwise.

References [Plane::can_land_on\(\)](#).

Here is the call graph for this function:



4.5.2.2 `exhaust_rate()` [1/2]

```
double Flight::exhaust_rate ( ) const
```

Calculates the workload for the pilot(s) during this flight.

Returns

The workload as a percentage.

4.5.2.3 exhaust_rate() [2/2]

```
double Flight::exhaust_rate (
    const Plane * _plane ) const
```

Calculates the workload for a given plane during this flight.

Parameters

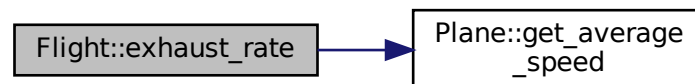
<code>_plane</code>	The plane to evaluate.
---------------------	------------------------

Returns

The workload as a percentage for the specified plane.

References [Plane::get_average_speed\(\)](#).

Here is the call graph for this function:



4.5.2.4 flight_duration() [1/2]

```
double Flight::flight_duration ( ) const
```

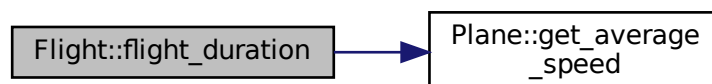
Calculates the flight duration based on the route length and plane speed.

Returns

The flight duration in hours.

References [Plane::get_average_speed\(\)](#).

Here is the call graph for this function:



4.5.2.5 `flight_duration()` [2/2]

```
double Flight::flight_duration (
    const Plane * _plane ) const
```

Calculates the flight duration based on the route length and plane speed.

Parameters

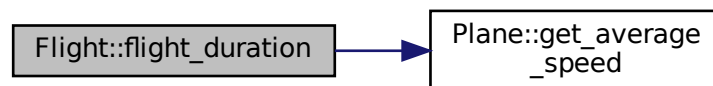
<code>_plane</code>	The plane to evaluate.
---------------------	------------------------

Returns

The flight duration with the given plane in hours.

References [Plane::get_average_speed\(\)](#).

Here is the call graph for this function:



4.5.2.6 `set_from()`

```
bool Flight::set_from (
    const Airport & from )
```

Sets the origin airport for the flight.

Parameters

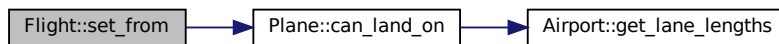
<code>from</code>	The airport object representing the flight's origin.
-------------------	--

Returns

True if the origin was set successfully, false otherwise.

References [Plane::can_land_on\(\)](#).

Here is the call graph for this function:



4.5.2.7 set_to()

```
bool Flight::set_to (
    const Airport & to )
```

Sets the destination airport for the flight.

Parameters

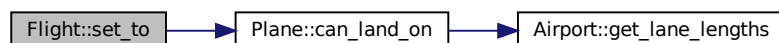
<code>to</code>	The airport object representing the flight's destination.
-----------------	---

Returns

True if the destination was set successfully, false otherwise.

References [Plane::can_land_on\(\)](#).

Here is the call graph for this function:



4.5.3 Friends And Related Function Documentation

4.5.3.1 AirSpaceManager

```
friend class AirSpaceManager [friend]
```

Declares [AirSpaceManager](#) as a friend class to access private members.

4.5.3.2 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Flight & flight ) [friend]
```

Overloaded output stream operator for [Flight](#).

Parameters

<i>out</i>	The output stream.
<i>flight</i>	The Flight object to be output.

Returns

Reference to the output stream.

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

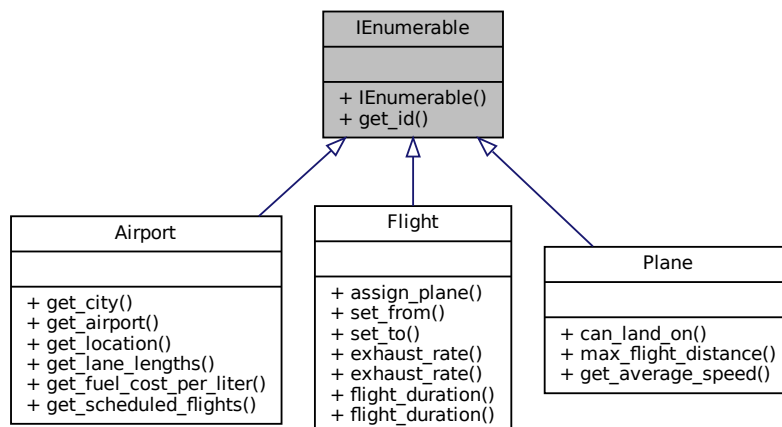
- [flight.hpp](#)
- [flight.cpp](#)

4.6 IEnumerable Class Reference

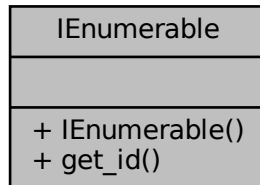
A base class that provides unique IDs for objects.

```
#include <enumerable.hpp>
```

Inheritance diagram for IEnumerable:



Collaboration diagram for IEnumerable:



Public Member Functions

- [IEnumerable](#) ()
Default constructor. Assigns a unique ID to the instance.
- virtual unsigned int [get_id](#) () const final
Retrieves the unique ID of the instance.

Friends

- class [AirSpaceManager](#)
Declares [AirSpaceManager](#) as a friend class, allowing it to access private members.

4.6.1 Detailed Description

A base class that provides unique IDs for objects.

This class is designed to assign a unique identifier to each instance of a derived class.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 IEnumerable()

```
IEnumerable::IEnumerable ( ) [inline]
```

Default constructor. Assigns a unique ID to the instance.

4.6.3 Member Function Documentation

4.6.3.1 get_id()

```
virtual unsigned int IEnumerable::get_id ( ) const [inline], [final], [virtual]
```

Retrieves the unique ID of the instance.

Returns

The unique ID of the object.

4.6.4 Friends And Related Function Documentation

4.6.4.1 AirSpaceManager

```
friend class AirSpaceManager [friend]
```

Declares [AirSpaceManager](#) as a friend class, allowing it to access private members.

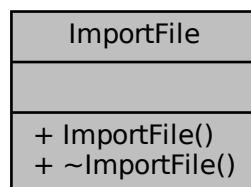
The documentation for this class was generated from the following files:

- [enumerable.hpp](#)
- [enumerable.cpp](#)

4.7 ImportFile Class Reference

```
#include <import_file.hpp>
```

Collaboration diagram for ImportFile:



Public Member Functions

- [ImportFile](#) (const std::string &)
- [~ImportFile](#) ()

4.7.1 Constructor & Destructor Documentation

4.7.1.1 ImportFile()

```
ImportFile::ImportFile (
    const std::string & filename )
```

4.7.1.2 ~ImportFile()

```
ImportFile::~ImportFile ( )
```

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

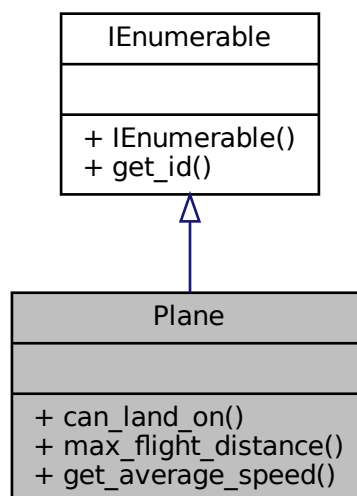
- [import_file.hpp](#)
- [import_file.cpp](#)

4.8 Plane Class Reference

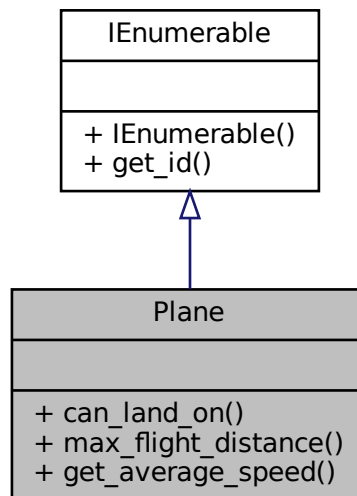
Represents an aircraft with details for efficient airline management.

```
#include <plane.hpp>
```

Inheritance diagram for Plane:



Collaboration diagram for Plane:



Public Member Functions

- bool `can_land_on` (const [Airport](#) &airport) const
Checks if the plane can land on a given airport.
- double `max_flight_distance` () const
Calculates the maximum flight distance for the plane.
- double `get_average_speed` () const
Retrieves the average speed of the plane.

Friends

- class [AirSpaceManager](#)
Declares [AirSpaceManager](#) as a friend class. Allows access to the private and protected members of [Plane](#).
- std::ostream & `operator<<` (std::ostream &out, const [Plane](#) &plane)
Overloaded output stream operator for [Plane](#). Outputs the details of the plane to the provided stream.

4.8.1 Detailed Description

Represents an aircraft with details for efficient airline management.

The [Plane](#) class encapsulates data about the manufacturer, model, seating capacity, minimum runway length, operational costs, fuel consumption, tank volume, and average speed.

4.8.2 Member Function Documentation

4.8.2.1 can_land_on()

```
bool Plane::can_land_on (
    const Airport & airport ) const
```

Checks if the plane can land on a given airport.

Parameters

<i>airport</i>	The airport to check compatibility with.
----------------	--

Returns

True if the plane can land, false otherwise.

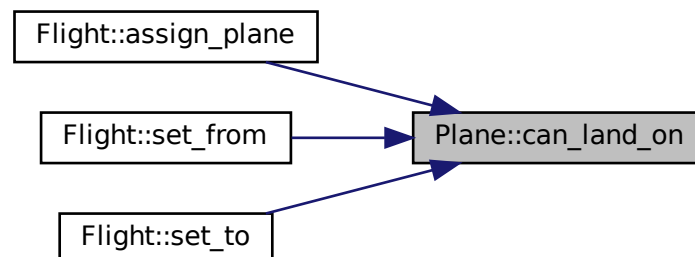
References [Airport::get_lane_lengths\(\)](#).

Referenced by [Flight::assign_plane\(\)](#), [Flight::set_from\(\)](#), and [Flight::set_to\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.2.2 `get_average_speed()`

```
double Plane::get_average_speed ( ) const
```

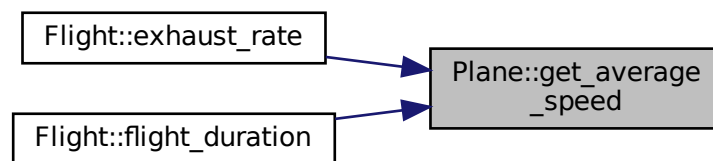
Retrieves the average speed of the plane.

Returns

The average speed in kilometers per hour.

Referenced by [Flight::exhaust_rate\(\)](#), and [Flight::flight_duration\(\)](#).

Here is the caller graph for this function:



4.8.2.3 `max_flight_distance()`

```
double Plane::max_flight_distance ( ) const
```

Calculates the maximum flight distance for the plane.

Returns

The maximum distance the plane can travel in kilometers.

4.8.3 Friends And Related Function Documentation

4.8.3.1 `AirSpaceManager`

```
friend class AirSpaceManager [friend]
```

Declares [AirSpaceManager](#) as a friend class. Allows access to the private and protected members of [Plane](#).

4.8.3.2 `operator<<`

```
std::ostream& operator<< (
    std::ostream & out,
    const Plane & plane ) [friend]
```

Overloaded output stream operator for [Plane](#). Outputs the details of the plane to the provided stream.

Parameters

<i>out</i>	The output stream.
<i>plane</i>	The plane object to be output.

Returns

Reference to the output stream.

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

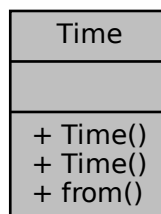
- [plane.hpp](#)
- [plane.cpp](#)

4.9 Time Class Reference

Represents a time duration or specific time of day without date information.

```
#include <date_time.hpp>
```

Collaboration diagram for Time:



Public Member Functions

- [Time](#) ()
Default constructor initializing to an undefined time.
- [Time](#) (unsigned short hour, unsigned short minute, unsigned short second)
Constructs a [Time](#) object with the specified hours, minutes, and seconds.

Static Public Member Functions

- static [Time](#) from (double hours)
Converts a time duration in hours (as a double) to a [Time](#) object.

Friends

- `std::ostream & operator<< (std::ostream &out, const Time &time)`
Overloaded output stream operator for *Time*.

4.9.1 Detailed Description

Represents a time duration or specific time of day without date information.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Time() [1/2]

```
Time::Time ( )
```

Default constructor initializing to an undefined time.

Referenced by [from\(\)](#).

Here is the caller graph for this function:



4.9.2.2 Time() [2/2]

```
Time::Time (
    unsigned short hour,
    unsigned short minute,
    unsigned short second )
```

Constructs a *Time* object with the specified hours, minutes, and seconds.

Parameters

<i>hour</i>	The hour of the time (0-23).
<i>minute</i>	The minute of the time (0-59).
<i>second</i>	The second of the time (0-59).

4.9.3 Member Function Documentation

4.9.3.1 from()

```
Time Time::from (
    double hours ) [static]
```

Converts a time duration in hours (as a double) to a [Time](#) object.

Parameters

<i>hours</i>	The time duration in hours.
--------------	-----------------------------

Returns

A [Time](#) object representing the duration.

References [Time\(\)](#).

Here is the call graph for this function:



4.9.4 Friends And Related Function Documentation

4.9.4.1 operator<<

```
std::ostream& operator<< (
    std::ostream & out,
    const Time & time ) [friend]
```

Overloaded output stream operator for [Time](#).

Parameters

<i>out</i>	The output stream.
<i>time</i>	The Time object to be output.

Returns

Reference to the output stream.

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

- [date_time.hpp](#)
- [date_time.cpp](#)

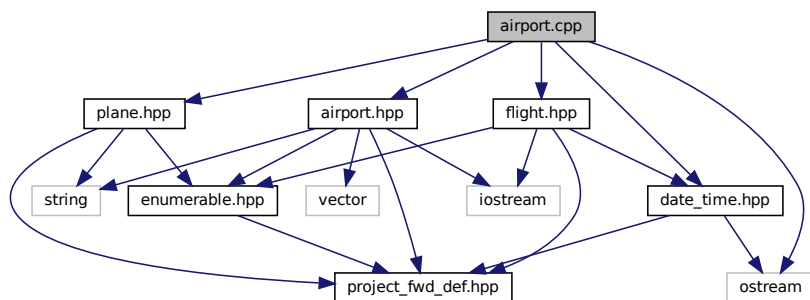
Chapter 5

File Documentation

5.1 airport.cpp File Reference

```
#include "airport.hpp"
#include "date_time.hpp"
#include "flight.hpp"
#include "plane.hpp"
#include <ostream>
```

Include dependency graph for airport.cpp:



Functions

- `std::ostream & operator<< (std::ostream &out, const Airport &airport)`

5.1.1 Function Documentation

5.1.1.1 operator<<()

```
std::ostream& operator<< (
    std::ostream & out,
    const Airport & airport )
```

Parameters

<i>out</i>	The output stream to write to.
<i>airport</i>	The Airport object to output.

Returns

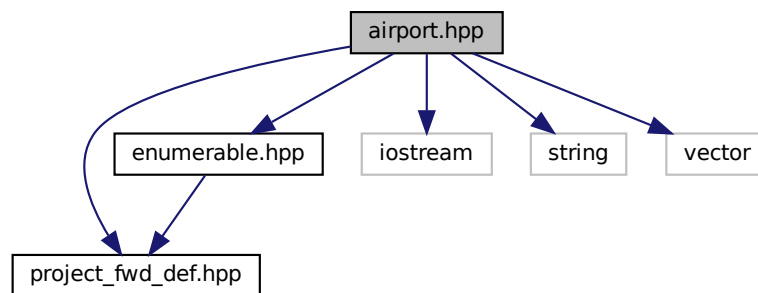
A reference to the output stream.

5.2 airport.hpp File Reference

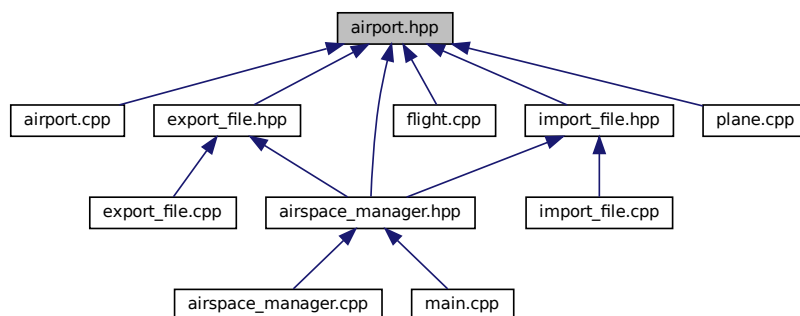
Contains the declaration of the [Airport](#) class and its methods.

```
#include "project_fwd_def.hpp"
#include "enumerable.hpp"
#include <iostream>
#include <string>
#include <vector>
```

Include dependency graph for airport.hpp:



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



Data Structures

- class [Airport](#)

Represents an airport with details such as city, name, location, lane lengths, fuel cost, and scheduled flights.

5.2.1 Detailed Description

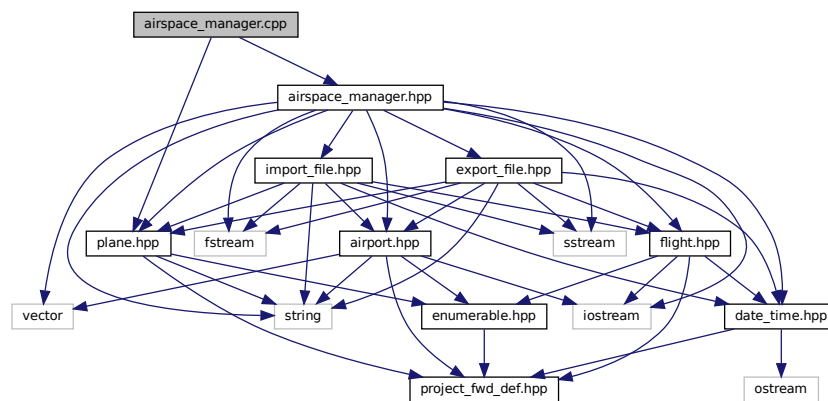
Contains the declaration of the [Airport](#) class and its methods.

5.3 airspace_manager.cpp File Reference

```
#include "airspace_manager.hpp"
```

```
#include "plane.hpp"
```

Include dependency graph for `airspace_manager.cpp`:



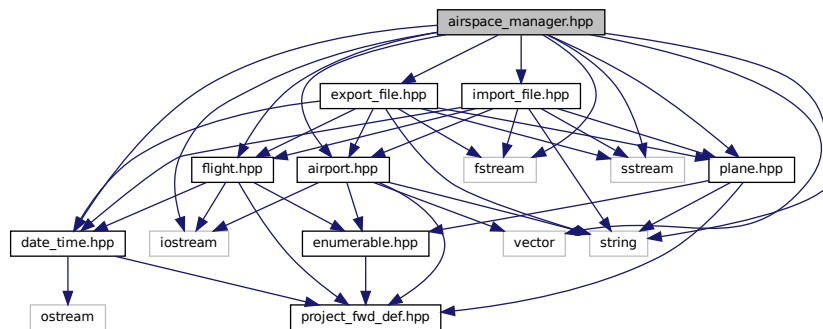
5.4 airspace_manager.hpp File Reference

Contains the declaration of the [AirSpaceManager](#) class and its methods for managing airports, planes, and flights.

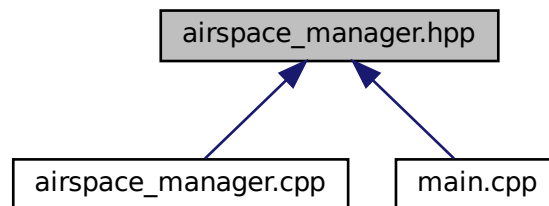
```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <vector>
#include "date_time.hpp"
#include "airport.hpp"
#include "plane.hpp"
#include "flight.hpp"
#include "export_file.hpp"
```

```
#include "import_file.hpp"
```

Include dependency graph for `airspace_manager.hpp`:



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



Data Structures

- class [AirSpaceManager](#)

A class to manage airports, planes, and flights for optimizing airline resources.

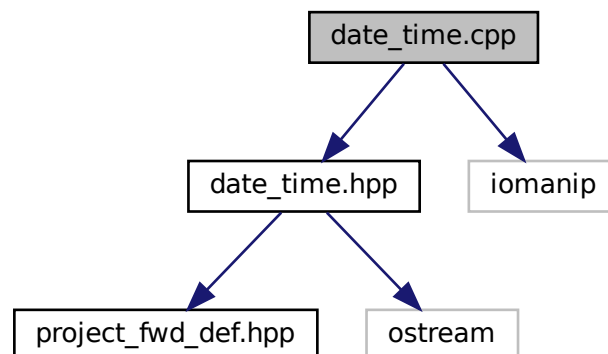
5.4.1 Detailed Description

Contains the declaration of the [AirSpaceManager](#) class and its methods for managing airports, planes, and flights.

5.5 date_time.cpp File Reference

```
#include "date_time.hpp"
#include <iomanip>
```

Include dependency graph for date_time.cpp:



Functions

- `std::ostream & operator<< (std::ostream &out, const DateTime &time)`
- `std::ostream & operator<< (std::ostream &out, const Time &time)`

5.5.1 Function Documentation

5.5.1.1 operator<<() [1/2]

```
std::ostream& operator<< (
    std::ostream & out,
    const DateTime & time )
```

Parameters

<i>out</i>	The output stream.
<i>time</i>	The <code>DateTime</code> object to be output.

Returns

Reference to the output stream.

5.5.1.2 operator<<() [2/2]

```
std::ostream& operator<< (
    std::ostream & out,
    const Time & time )
```

Parameters

<i>out</i>	The output stream.
<i>time</i>	The Time object to be output.

Returns

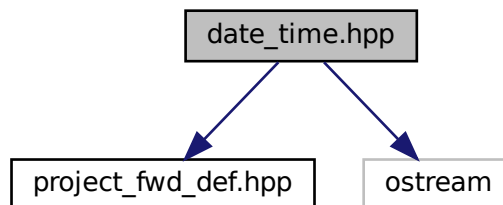
Reference to the output stream.

5.6 date_time.hpp File Reference

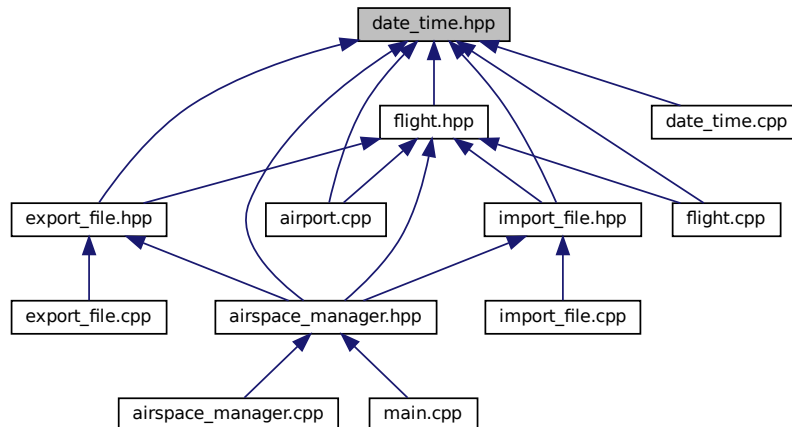
Contains the declaration of the [DateTime](#) and [Time](#) classes and their methods for handling date and time information.

```
#include "project_fwd_def.hpp"
#include <ostream>
```

Include dependency graph for date_time.hpp:



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



Data Structures

- class [DateTime](#)
Represents a specific point in time with detailed components like year, month, day, hour, minute, and second.
- class [Time](#)
Represents a time duration or specific time of day without date information.

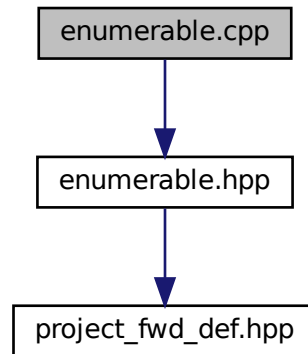
5.6.1 Detailed Description

Contains the declaration of the [DateTime](#) and [Time](#) classes and their methods for handling date and time information.

5.7 enumerable.cpp File Reference

```
#include "enumerable.hpp"
```

Include dependency graph for enumerable.cpp:

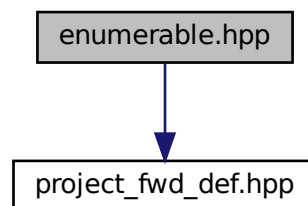


5.8 enumerable.hpp File Reference

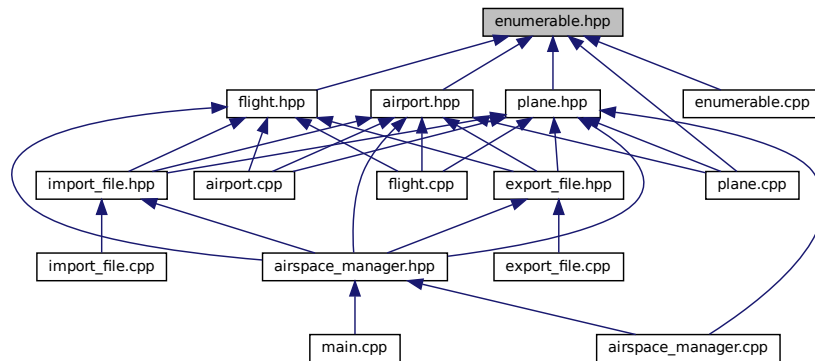
Contains the declaration of the [IEnumerable](#) class, which provides a unique identifier for derived objects.

```
#include "project_fwd_def.hpp"
```

Include dependency graph for enumerable.hpp:



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



Data Structures

- class [IEnumerable](#)

A base class that provides unique IDs for objects.

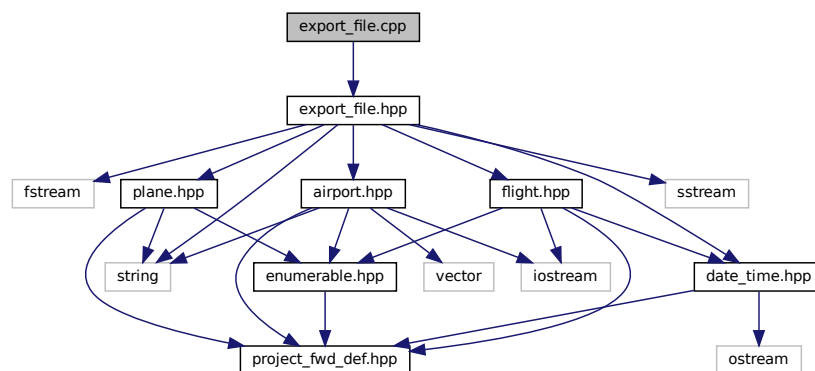
5.8.1 Detailed Description

Contains the declaration of the [IEnumerable](#) class, which provides a unique identifier for derived objects.

5.9 export_file.cpp File Reference

```
#include "export_file.hpp"
```

Include dependency graph for export_file.cpp:

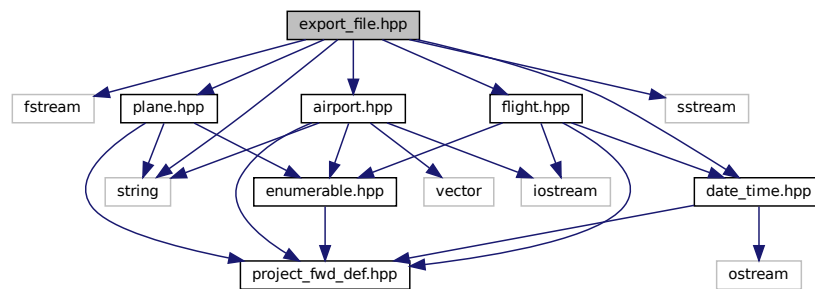


5.10 export_file.hpp File Reference

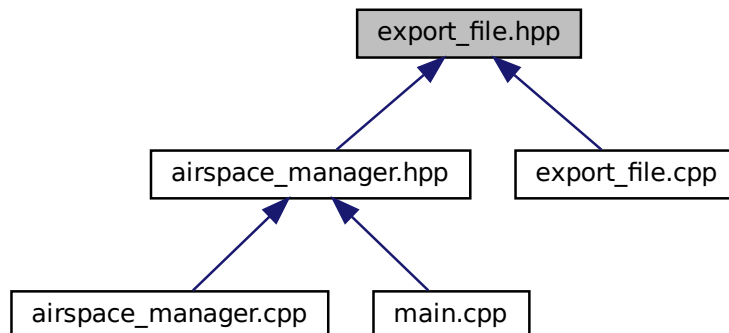
Contains a wrapper for export file.

```
#include <fstream>
#include <string>
#include <sstream>
#include "date_time.hpp"
#include "airport.hpp"
#include "plane.hpp"
#include "flight.hpp"
```

Include dependency graph for export_file.hpp:



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



Data Structures

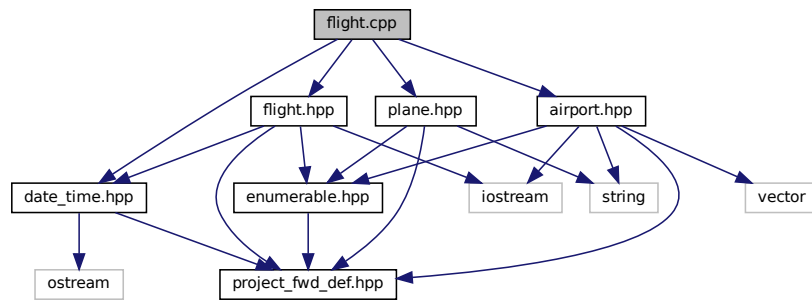
- class [ExportFile](#)

5.10.1 Detailed Description

Contains a wrapper for export file.

5.11 flight.cpp File Reference

```
#include "airport.hpp"
#include "date_time.hpp"
#include "flight.hpp"
#include "plane.hpp"
Include dependency graph for flight.cpp:
```



Functions

- `std::ostream & operator<< (std::ostream &out, const Flight &flight)`

5.11.1 Function Documentation

5.11.1.1 operator<<()

```
std::ostream& operator<< (
    std::ostream & out,
    const Flight & flight )
```

Parameters

<i>out</i>	The output stream.
<i>flight</i>	The Flight object to be output.

Returns

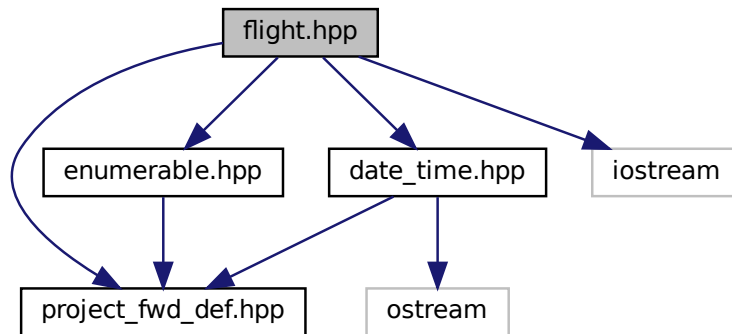
Reference to the output stream.

5.12 flight.hpp File Reference

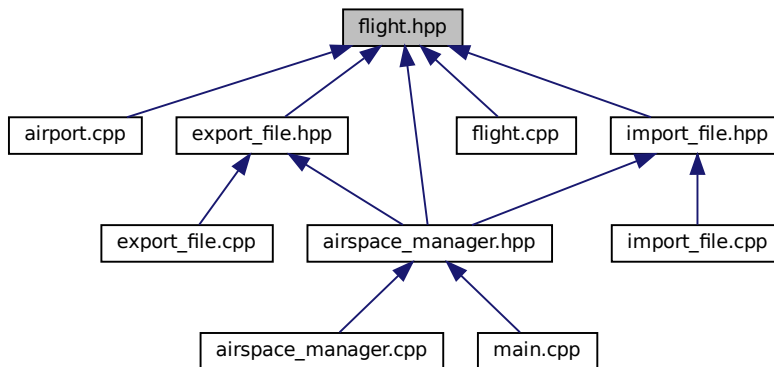
Contains the declaration of the [Flight](#) class, representing a flight with associated data and operations.

```
#include "project_fwd_def.hpp"
#include "date_time.hpp"
#include "enumerable.hpp"
#include <iostream>
```

Include dependency graph for flight.hpp:



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



Data Structures

- class [Flight](#)

Represents a flight, including its schedule, route, and assigned plane.

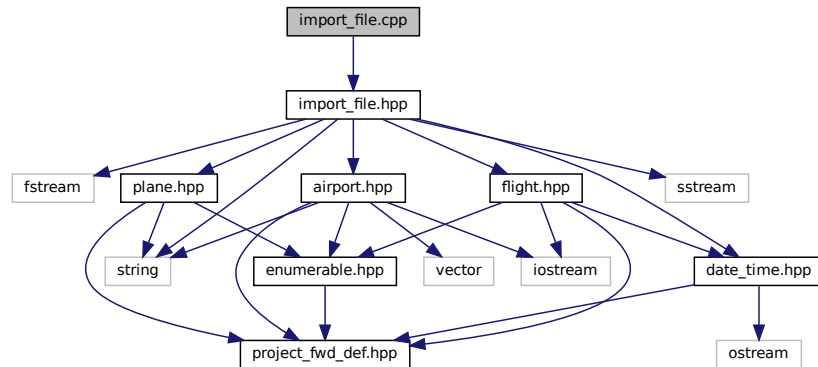
5.12.1 Detailed Description

Contains the declaration of the [Flight](#) class, representing a flight with associated data and operations.

5.13 import_file.cpp File Reference

```
#include "import_file.hpp"
```

Include dependency graph for import_file.cpp:

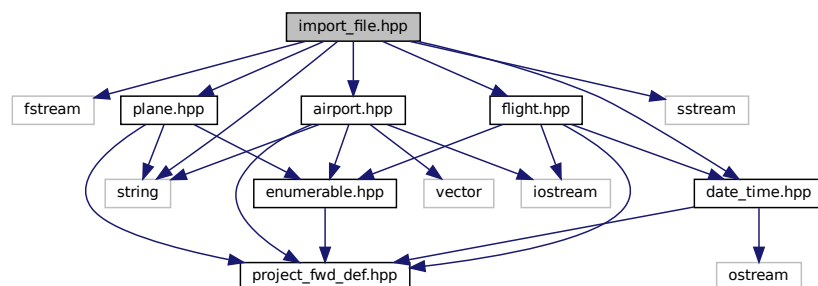


5.14 import_file.hpp File Reference

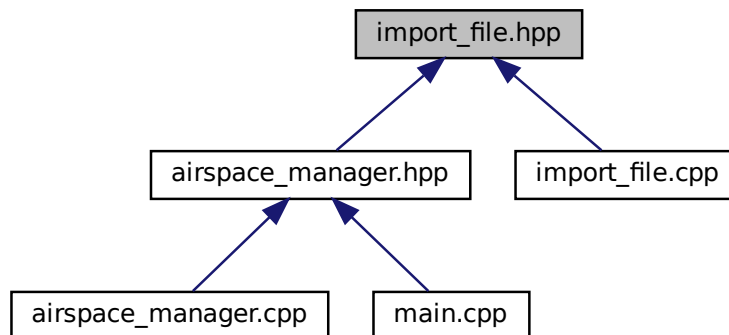
Contains a wrapper for import file.

```
#include <fstream>
#include <string>
#include <sstream>
#include "date_time.hpp"
#include "airport.hpp"
#include "plane.hpp"
#include "flight.hpp"
```

Include dependency graph for import_file.hpp:



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



Data Structures

- class [ImportFile](#)

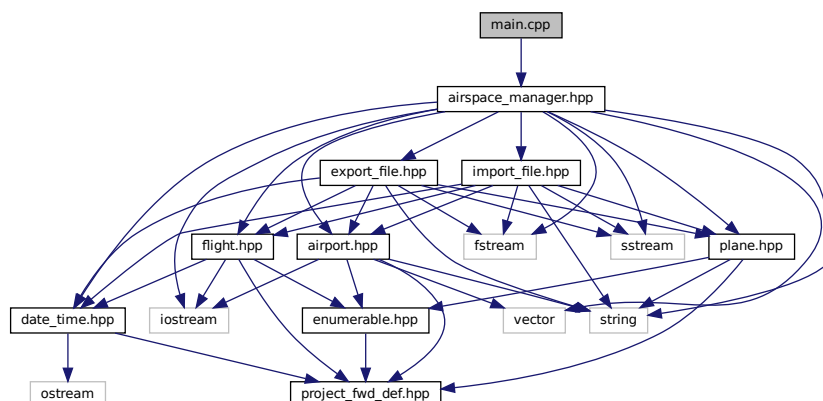
5.14.1 Detailed Description

Contains a wrapper for import file.

5.15 main.cpp File Reference

```
#include "airspace_manager.hpp"
```

Include dependency graph for `main.cpp`:



Functions

- int [main](#) ()

5.15.1 Function Documentation

5.15.1.1 main()

```
int main ( )
```

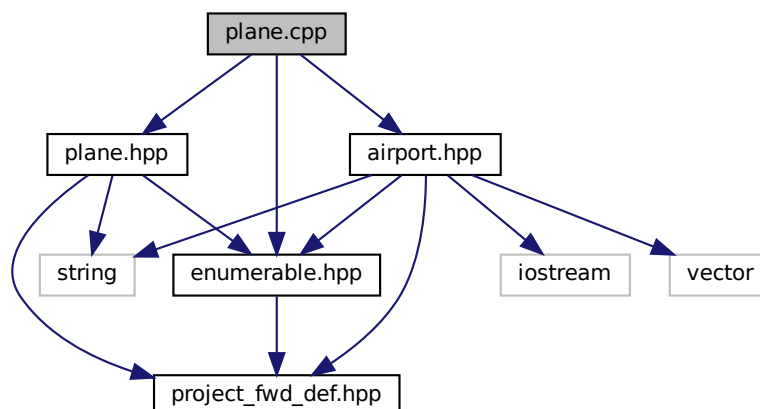
References [AirSpaceManager::run\(\)](#).

Here is the call graph for this function:



5.16 plane.cpp File Reference

```
#include "airport.hpp"
#include "enumerable.hpp"
#include "plane.hpp"
Include dependency graph for plane.cpp:
```



Functions

- `std::ostream & operator<< (std::ostream &out, const Plane &plane)`

5.16.1 Function Documentation

5.16.1.1 `operator<<()`

```
std::ostream& operator<< (
    std::ostream & out,
    const Plane & plane )
```

Parameters

<i>out</i>	The output stream.
<i>plane</i>	The plane object to be output.

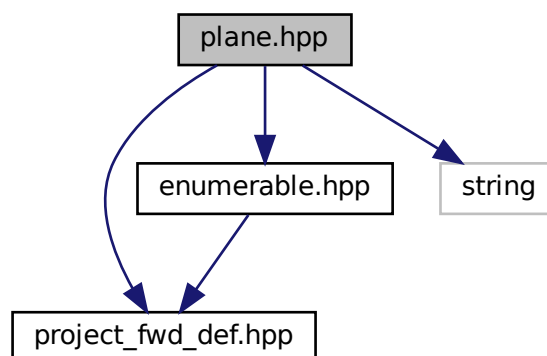
Returns

Reference to the output stream.

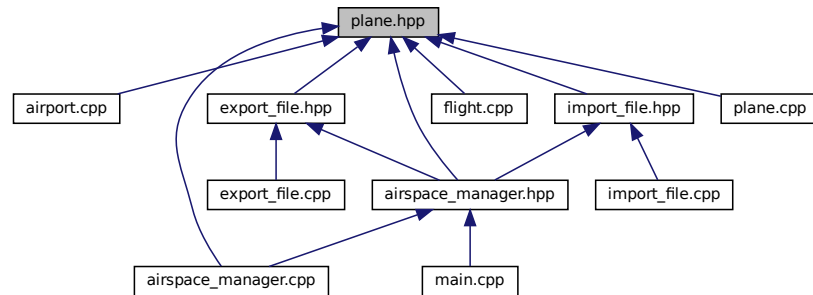
5.17 `plane.hpp` File Reference

Contains the declaration of the [Plane](#) class and its methods.

```
#include "project_fwd_def.hpp"
#include "enumerable.hpp"
#include <string>
Include dependency graph for plane.hpp:
```



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



Data Structures

- class [Plane](#)

Represents an aircraft with details for efficient airline management.

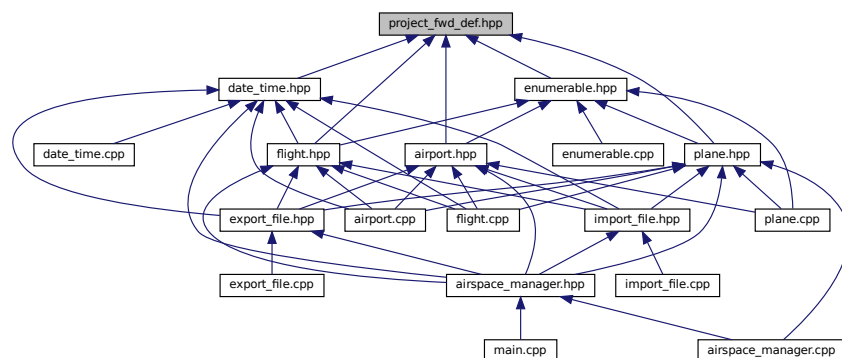
5.17.1 Detailed Description

Contains the declaration of the [Plane](#) class and its methods.

5.18 project_fwd_def.hpp File Reference

Contains the declaration of the `project_fwd_def` class and its methods.

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



5.18.1 Detailed Description

Contains the declaration of the `project_fwd_def` class and its methods.

Index

- ~AirSpaceManager
 - AirSpaceManager, 12
- ~ExportFile
 - ExportFile, 16
- ~ImportFile
 - ImportFile, 26
- Airport, 7
 - AirSpaceManager, 11
 - get_airport, 9
 - get_city, 9
 - get_fuel_cost_per_liter, 9
 - get_lane_lengths, 9
 - get_location, 10
 - get_scheduled_flights, 10
 - operator<<, 11
- airport.cpp, 35
 - operator<<, 35
- airport.hpp, 36
- airspace_manager.cpp, 37
- airspace_manager.hpp, 37
- AirSpaceManager, 11
 - ~AirSpaceManager, 12
 - Airport, 11
 - AirSpaceManager, 12
 - DateTime, 15
 - Flight, 22
 - IEnumerable, 25
 - input_airport, 13
 - input_flight, 13
 - input_plane, 13
 - Plane, 29
 - run, 13
- assign_plane
 - Flight, 19
- can_land_on
 - Plane, 27
- date_time.cpp, 38
 - operator<<, 39
- date_time.hpp, 40
- DateTime, 14
 - AirSpaceManager, 15
 - DateTime, 14
 - operator<<, 15
- enumerable.cpp, 42
- enumerable.hpp, 42
- exhaust_rate
 - Flight, 19
- export_file.cpp, 43
- export_file.hpp, 44
- ExportFile, 16
 - ~ExportFile, 16
 - ExportFile, 16
- Flight, 17
 - AirSpaceManager, 22
 - assign_plane, 19
 - exhaust_rate, 19
 - flight_duration, 20
 - operator<<, 22
 - set_from, 21
 - set_to, 22
- flight.cpp, 45
 - operator<<, 45
- flight.hpp, 45
- flight_duration
 - Flight, 20
- from
 - Time, 32
- get_airport
 - Airport, 9
- get_average_speed
 - Plane, 28
- get_city
 - Airport, 9
- get_fuel_cost_per_liter
 - Airport, 9
- get_id
 - IEnumerable, 24
- get_lane_lengths
 - Airport, 9
- get_location
 - Airport, 10
- get_scheduled_flights
 - Airport, 10
- IEnumerable, 23
 - AirSpaceManager, 25
 - get_id, 24
 - IEnumerable, 24
- import_file.cpp, 47
- import_file.hpp, 47
- ImportFile, 25
 - ~ImportFile, 26
 - ImportFile, 26
- input_airport

- AirSpaceManager, [13](#)
- input_flight
 - AirSpaceManager, [13](#)
- input_plane
 - AirSpaceManager, [13](#)
- main
 - main.cpp, [49](#)
- main.cpp, [48](#)
 - main, [49](#)
- max_flight_distance
 - Plane, [29](#)
- operator<<
 - Airport, [11](#)
 - airport.cpp, [35](#)
 - date_time.cpp, [39](#)
 - DateTime, [15](#)
 - Flight, [22](#)
 - flight.cpp, [45](#)
 - Plane, [29](#)
 - plane.cpp, [50](#)
 - Time, [32](#)
- Plane, [26](#)
 - AirSpaceManager, [29](#)
 - can_land_on, [27](#)
 - get_average_speed, [28](#)
 - max_flight_distance, [29](#)
 - operator<<, [29](#)
- plane.cpp, [49](#)
 - operator<<, [50](#)
- plane.hpp, [50](#)
- project_fwd_def.hpp, [51](#)
- run
 - AirSpaceManager, [13](#)
- set_from
 - Flight, [21](#)
- set_to
 - Flight, [22](#)
- Time, [30](#)
 - from, [32](#)
 - operator<<, [32](#)
 - Time, [31](#)