Документация на проекта за управление на полети и самолети

Алекс Иванов Цветанов, 38^{ма} група, КСИ, ФКСТ, №121222225

3 декември 2024 г.

Глава 1

Основи на проекта

1.1 Въведение

Проектът е разработен с цел да подпомогне авиокомпании при управлението на ресурси чрез оптимизиране на самолетните полети. Целите на проекта включват:

- Съхранение на информация за самолети и полети.
- Извършване на търсения за съвместимост между самолети и дестинации.
- Оценка на ефективността и ресурсоемкостта на даден самолетен маршрут.

1.2 Файлова структура

Проектът съдържа следните основни файлове:

- main.cpp: Основният файл за изпълнение.
- airport.cpp, airport.hpp: Модул за управление на летища.
- airspace_manager.cpp, airspace_manager.hpp: Логика за управление на въздушното пространство.
- plane.cpp, plane.hpp: Информация и операции, свързани със самолети.
- flight.cpp, flight.hpp: Управление на полети.
- date_time.cpp, date_time.hpp: Помощни функции за работа с дата и час.
- enumerable.cpp, enumerable.hpp: Общи структури за работа с итерации.
- CMakeLists.txt: Конфигурация на проекта.
- Doxyfile: Настройки за генериране на документация чрез Doxygen.
- description.txt: Текстово описание на проекта.

1.3 Описание на модулите

1.3.1 Модул Plane

Модулът управлява информацията за самолетите. Той съдържа следния клас:

Клас Plane

Полетата на класа:

- std::string id: Идентификационен номер на самолета.
- std::string manufacturer: Производител на самолета.
- std::string model: Модел на самолета.
- int seats: Брой седалки.
- double runway_length: Минимална дължина на пистата за кацане.
- double fuel_consumption: Разход на гориво за 1 км на място.
- double tank_capacity: Обем на резервоара.
- double average_speed: Средна скорост.

Член-функции:

- Plane(const std::string&, const std::string&, const std::string&, int, double): Конструктор за инициализация.
- void setId(const std::string&): Задава идентификационен номер.
- std::string getId() const: Връща идентификационния номер.
- void printDetails() const: Извежда информация за самолета.
- bool canReachDestination(double distance, double runway) const: Проверява съвместимостта с дадена дестинация.

1.3.2 Модул Flight

Модулът управлява полетите. Той съдържа следния клас:

Клас Flight

Полетата на класа:

- std::string flight_id: Идентификатор на полета.
- std::string destination: Дестинация.
- double distance: Разстояние на полета.
- Plane assigned_plane: Назначен самолет.

Член-функции:

- Flight(const std::string&, const std::string&, double): Конструктор за инициализация.
- void assignPlane(const Plane&): Назначава самолет към полета.
- bool isCompatible(const Plane&) const: Проверява дали самолетът е съвместим с полета.
- void printFlightDetails() const: Извежда информация за полета.

1.3.3 Модул AirspaceManager

Модулът управлява съвместимостта между самолети и дестинации. Той съдържа следния клас:

Клас AirspaceManager

Полетата на класа:

- std::vector<Plane> planes: Списък със самолети.
- std::vector<Flight> flights: Списък с полети.

Член-функции:

- void addPlane(const Plane&): Добавя самолет към списъка.
- void addFlight(const Flight&): Добавя полет към списъка.
- std::vector<Plane> findCompatiblePlanes(const Flight&) const: Връща списък с подходящи самолети за даден полет.
- void printAllFlights() const: Извежда информация за всички полети.
- void printAllPlanes() const: Извежда информация за всички самолети.

1.3.4 Модул DateTime

Модул за управление на дата и час. Той съдържа следния клас:

Клас DateTime

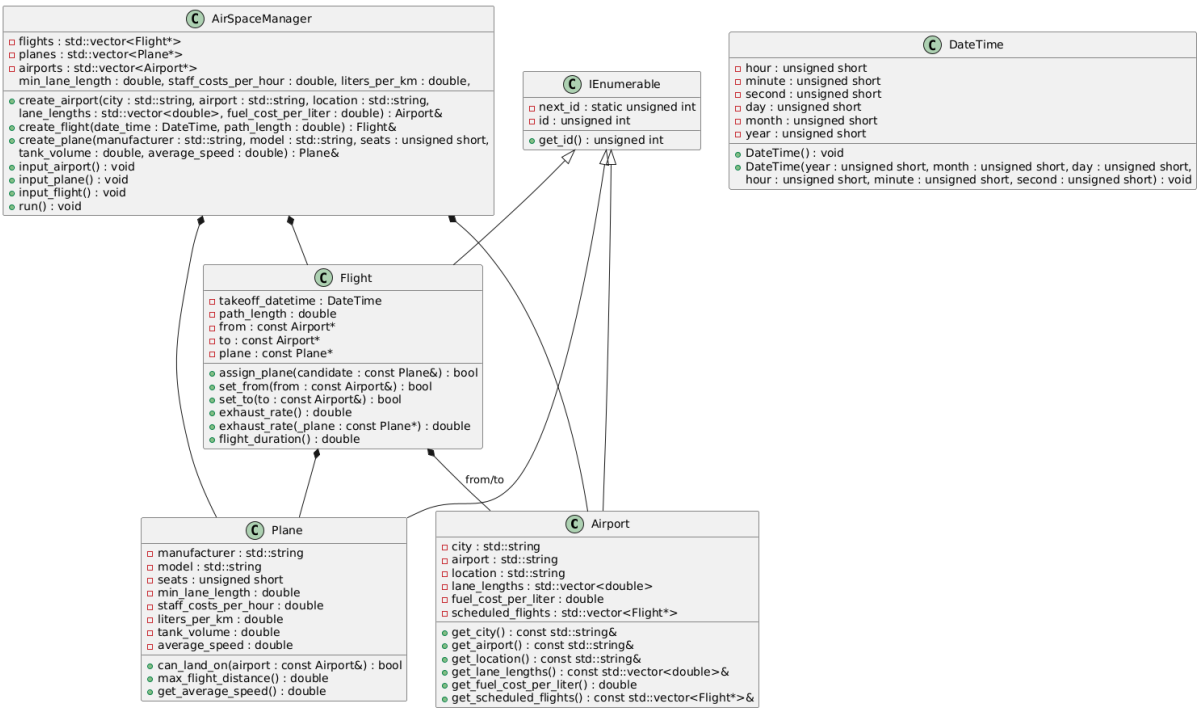
Полетата на класа:

- int year, month, day, hour, minute, second: Компоненти на дата и час.

Член-функции:

- DateTime(int, int, int, int, int, int): Конструктор за инициализация.
- std::string toString() const: Връща датата и часа като текст.
- bool isBefore(const DateTime&) const: Проверява дали една дата предхожда друга.

1.4 UML Диаграма



Глава 2

Hierarchical Index

2.1 Class Hierarchy

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

Глава 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Глава 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Глава 5

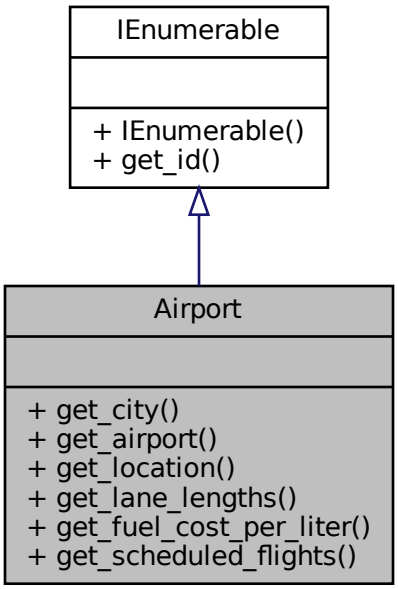
Data Structure Documentation

5.1 Airport Class Reference

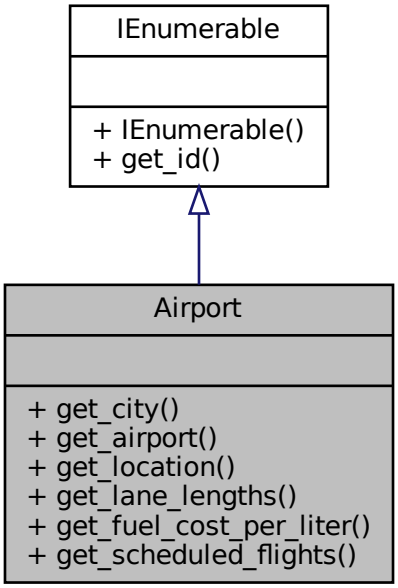
Represents an airport with details such as city, name, location, lane lengths, fuel cost, and scheduled flights.

#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.

5.1.1 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.

5.1.2 Member Function Documentation

5.1.2.1 get_airport()

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

Gets the name of the airport.

Returns

A constant reference to the airport name.

5.1.2.2 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.

5.1.2.3 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.

5.1.2.4 get_lane_lengths()

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

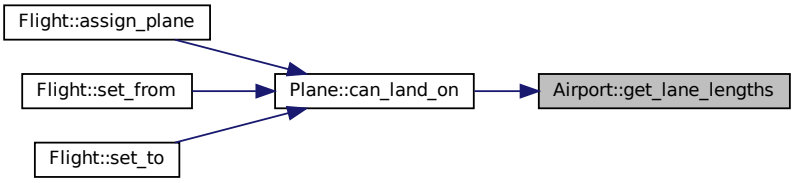
Gets the lengths of the lanes at the airport.

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:



5.1.2.5 get_location()

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

Gets the geographical location of the airport.

Returns

A constant reference to the location.

5.1.2.6 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.

5.1.3 Friends And Related Function Documentation

5.1.3.1 AirSpaceManager

friend class AirSpaceManager [friend]

Allows AirSpaceManager to access private members of the Airport class.

5.1.3.2 operator<<

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

Outputs the details of the Airport object to an output stream.

Parameters

out	The output stream to write to.
airport	The Airport object to output.

Returns

A reference to the output stream.

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

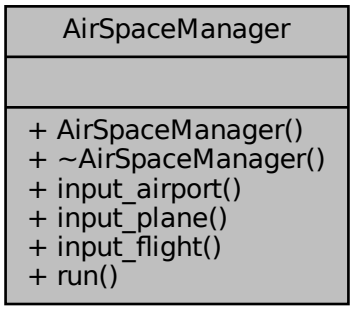
- airport.hpp
- airport.cpp

5.2 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.

5.2.1 Detailed Description

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

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AirSpaceManager()

AirSpaceManager::AirSpaceManager ( )

Default constructor for AirSpaceManager.

5.2.2.2 ~AirSpaceManager()

AirSpaceManager::~AirSpaceManager ( )

Destructor for AirSpaceManager. Frees allocated memory.

5.2.3 Member Function Documentation

5.2.3.1 input_airport()

void AirSpaceManager::input_airport ( )

Prompts the user to input details for a new airport.

5.2.3.2 input_flight()

void AirSpaceManager::input_flight ( )

Prompts the user to input details for a new flight.

5.2.3.3 input_plane()

void AirSpaceManager::input_plane ( )

Prompts the user to input details for a new plane.

5.2.3.4 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:

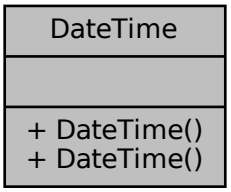
- airspace_manager.hpp
- airspace_manager.cpp

5.3 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

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

  Overloaded output stream operator for DateTime.

5.3.1 Detailed Description

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

5.3.2 Constructor & Destructor Documentation

5.3.2.1 DateTime() [1/2]

DateTime::DateTime ( )

Default constructor initializing to an undefined date and time.

5.3.2.2 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

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

5.3.3 Friends And Related Function Documentation

5.3.3.1 operator<<

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

Overloaded output stream operator for DateTime.

Parameters

out	The output stream.
time	The DateTime object to be output.

Returns

Reference to the output stream.

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

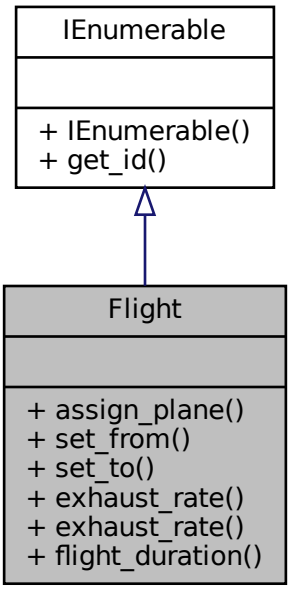
- date_time.hpp
- date_time.cpp

5.4 Flight Class Reference

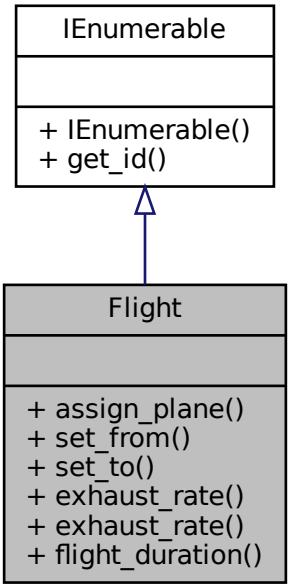
Represents a flight, including its schedule, route, and assigned plane.

#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.

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.

5.4.1 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.

5.4.2 Member Function Documentation

5.4.2.1 assign_plane()

bool Flight::assign_plane (
const Plane & candidate)

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

Parameters

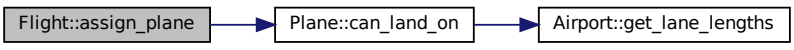
candidate	The plane to be considered for assignment.

Returns

True if the plane was successfully assigned, false otherwise.

References Plane::can_land_on().

Here is the call graph for this function:



5.4.2.2 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.

5.4.2.3 exhaust_rate() [2/2]

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

Calculates the workload for a given plane during this flight.

Parameters

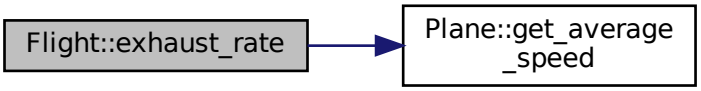
_plane	The plane to evaluate.

Returns

The workload as a percentage for the specified plane.

References Plane::get_average_speed().

Here is the call graph for this function:



5.4.2.4 flight_duration()

double Flight::flight_duration ( ) const

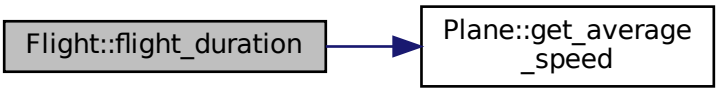
Calculates the flight duration based on the route length and plane speed.

Returns

The flight duration in hours.

References Plane::get_average_speed().

Here is the call graph for this function:



5.4.2.5 set_from()

bool Flight::set_from (
const Airport & from)

Sets the origin airport for the flight.

Parameters

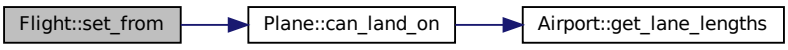
from	The airport object representing the flight's origin.

Returns

True if the origin was set successfully, false otherwise.

References Plane::can_land_on().

Here is the call graph for this function:



5.4.2.6 set_to()

bool Flight::set_to (
const Airport & to)

Sets the destination airport for the flight.

Parameters

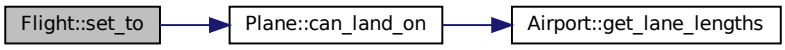
to	The airport object representing the flight's destination.

Returns

True if the destination was set successfully, false otherwise.

References Plane::can_land_on().

Here is the call graph for this function:



5.4.3 Friends And Related Function Documentation

5.4.3.1 AirSpaceManager

friend class AirSpaceManager [friend]

Declares AirSpaceManager as a friend class to access private members.

5.4.3.2 operator<<

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

Overloaded output stream operator for Flight.

Parameters

out	The output stream.
flight	The Flight object to be output.

Returns

Reference to the output stream.

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

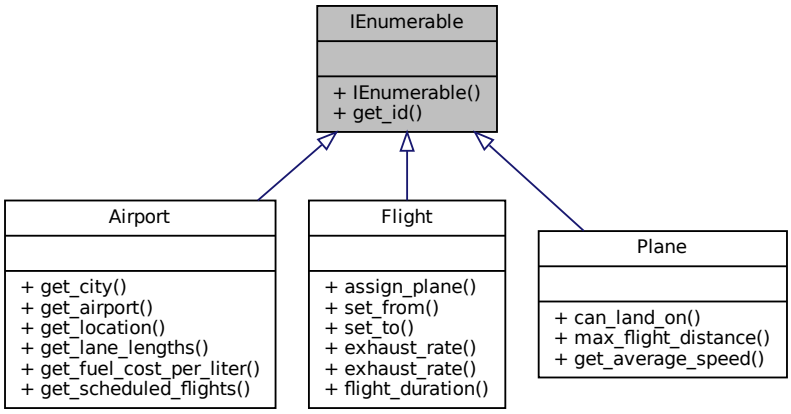
- flight.hpp
- flight.cpp

5.5 IEnumerable Class Reference

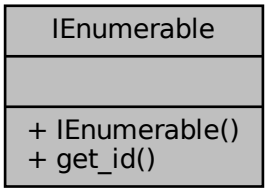
A base class that provides unique IDs for objects.

#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.

5.5.1 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.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 IEnumerable()

IEnumerable::IEnumerable ( ) [inline]

Default constructor. Assigns a unique ID to the instance.

5.5.3 Member Function Documentation

5.5.3.1 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.

5.5.4 Friends And Related Function Documentation

5.5.4.1 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:

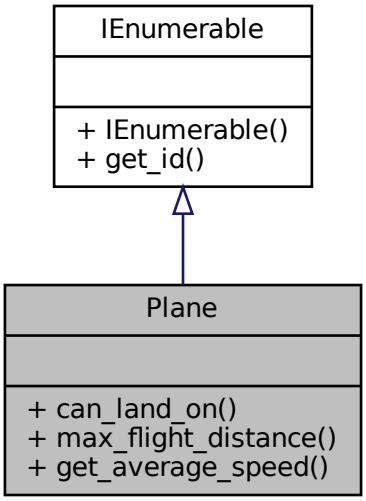
- enumerable.hpp
- enumerable.cpp

5.6 Plane Class Reference

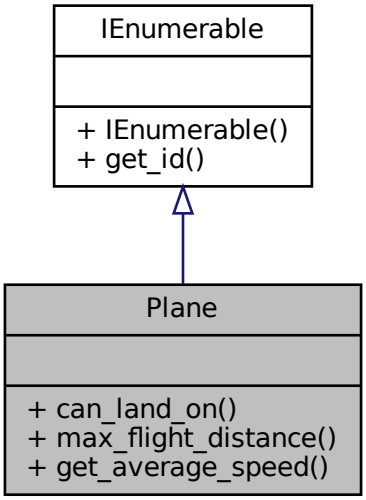
Represents an aircraft with details for efficient airline management.

#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.

5.6.1 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.

5.6.2 Member Function Documentation

5.6.2.1 can_land_on()

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

Checks if the plane can land on a given airport.

Parameters

airport	The airport to check compatibility with.

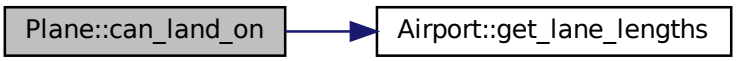
Returns

True if the plane can land, false otherwise.

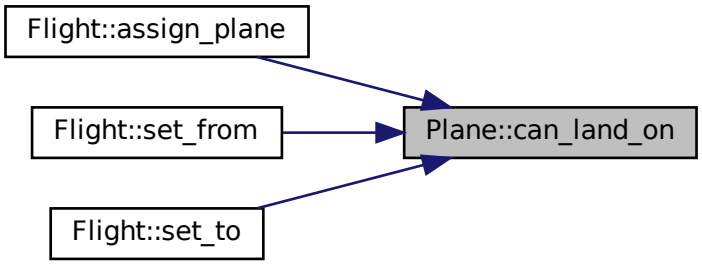
References Airport::get_lane_lengths().

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:



5.6.2.2 get_average_speed()

double Plane::get_average_speed ( ) const

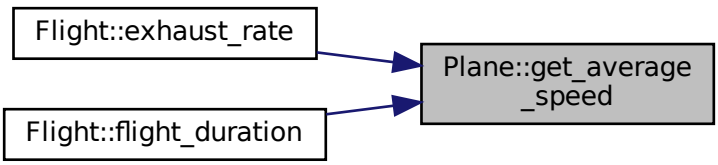
Retrieves the average speed of the plane.

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:



5.6.2.3 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.

5.6.3 Friends And Related Function Documentation

5.6.3.1 AirSpaceManager

friend class AirSpaceManager [friend]

Declares AirSpaceManager as a friend class. Allows access to the private and protected members of Plane.

5.6.3.2 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

out	The output stream.
plane	The plane object to be output.

Returns

Reference to the output stream.

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

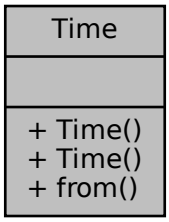
- plane.hpp
- plane.cpp

5.7 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.

5.7.1 Detailed Description

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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Time() [1/2]

Time::Time ( )

Default constructor initializing to an undefined time.

Referenced by from().

Here is the caller graph for this function:

5.7.2.2 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

hour	The hour of the time (0-23).
minute	The minute of the time (0-59).
second	The second of the time (0-59).

5.7.3 Member Function Documentation

5.7.3.1 from()

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

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

Parameters

hours	The time duration in hours.

Returns

A Time object representing the duration.

References Time().

Here is the call graph for this function:

5.7.4 Friends And Related Function Documentation

5.7.4.1 operator<<

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

Overloaded output stream operator for Time.

Parameters

out	The output stream.
time	The Time object to be output.

Returns

Reference to the output stream.

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

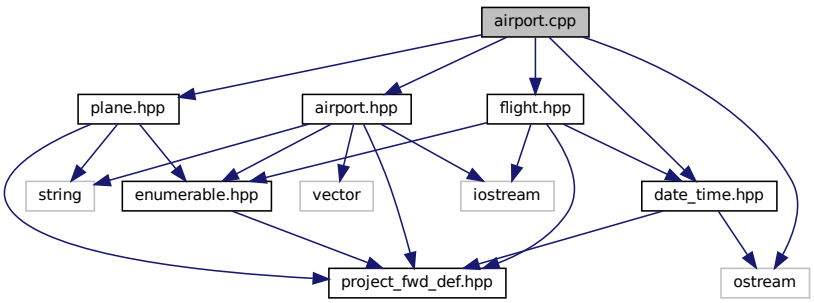
- date_time.hpp
- date_time.cpp

Глава 6

File Documentation

6.1 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)

6.1.1 Function Documentation

6.1.1.1 operator<<()

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

Parameters

out	The output stream to write to.
airport	The Airport object to output.

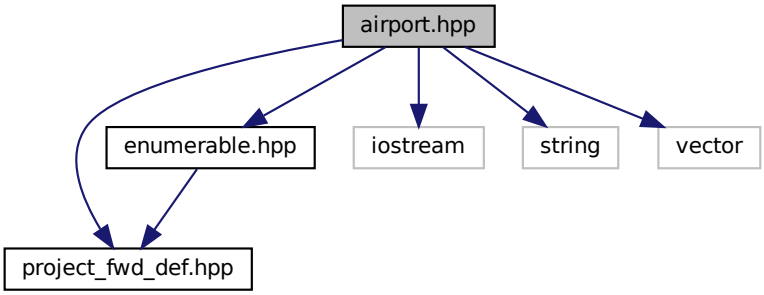
Returns

A reference to the output stream.

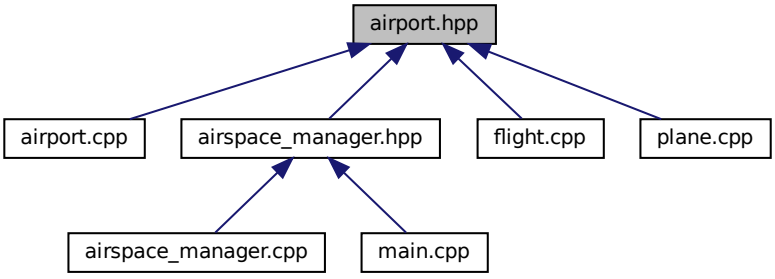
6.2 airport.hpp File Reference

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

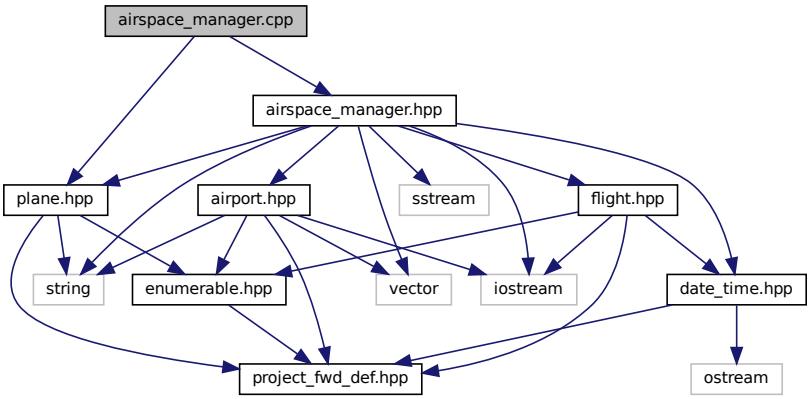
  Represents an airport with details such as city, name, location, lane lengths, fuel cost, and scheduled flights.

6.2.1 Detailed Description

Contains the declaration of the Airport class and its methods.

6.3 airspace_manager.cpp File Reference

#include "airspace_manager.hpp"
#include "plane.hpp"
Include dependency graph for airspace_manager.cpp:

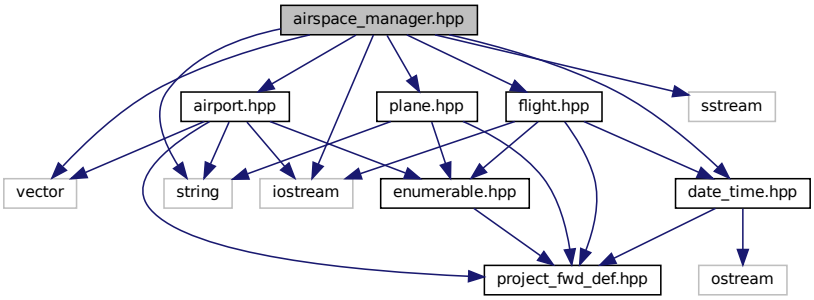


6.4 airspace_manager.hpp File Reference

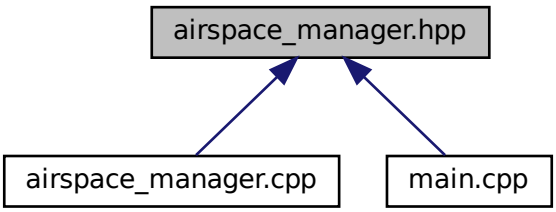
Contains the declaration of the AirSpaceManager class and its methods for managing airports, planes, and flights.

#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include "date_time.hpp"
#include "airport.hpp"
#include "plane.hpp"

#include "flight.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.

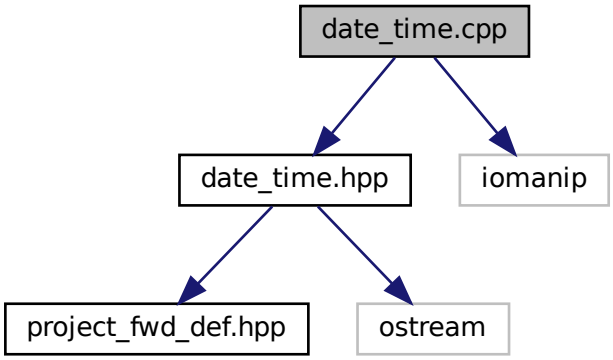
6.4.1 Detailed Description

Contains the declaration of the AirSpaceManager class and its methods for managing airports, planes, and flights.

6.5 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)

6.5.1 Function Documentation

6.5.1.1 operator<<() [1/2]

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

Parameters

out	The output stream.
time	The DateTime object to be output.

Returns

Reference to the output stream.

6.5.1.2 operator<<() [2/2]

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

Parameters

out	The output stream.
time	The Time object to be output.

Returns

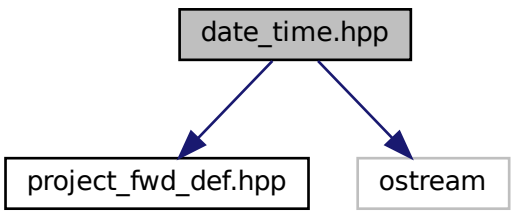
Reference to the output stream.

6.6 date_time.hpp File Reference

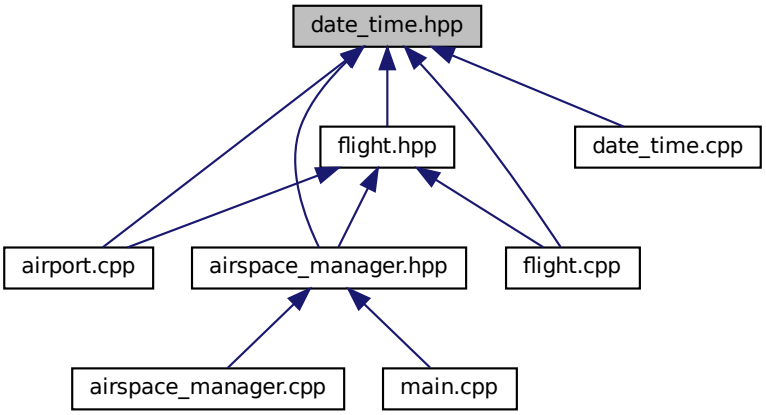
Contains the declaration of the DateTime and Time classes and their methods for handling date and time information.

```
#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

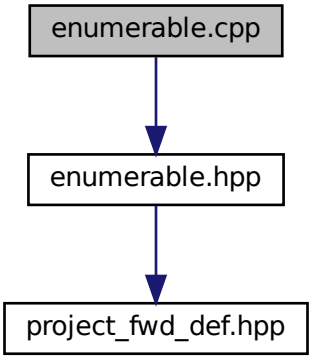
  Represents a time duration or specific time of day without date information.

6.6.1 Detailed Description

Contains the declaration of the DateTime and Time classes and their methods for handling date and time information.

6.7 enumerable.cpp File Reference

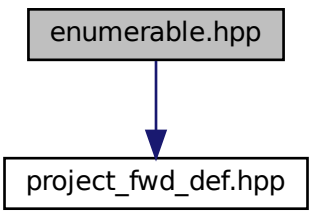
#include "enumerable.hpp"
Include dependency graph for enumerable.cpp:



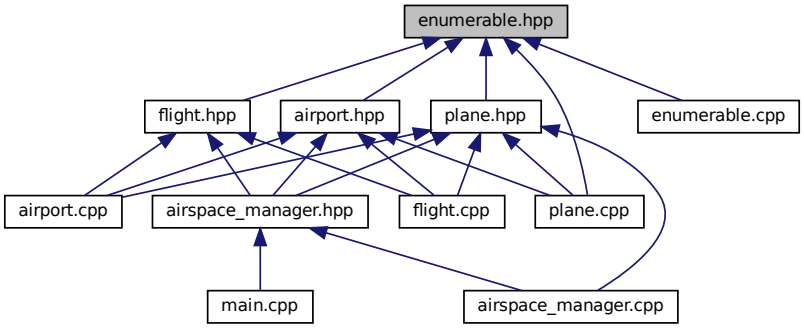
6.8 enumerable.hpp File Reference

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

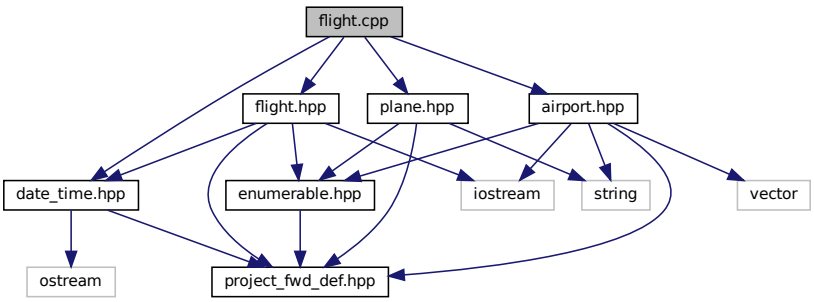
  A base class that provides unique IDs for objects.

6.8.1 Detailed Description

Contains the declaration of the IEnumerable class, which provides a unique identifier for derived objects.

6.9 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)

6.9.1 Function Documentation

6.9.1.1 operator<<()

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

Parameters

out	The output stream.
flight	The Flight object to be output.

Returns

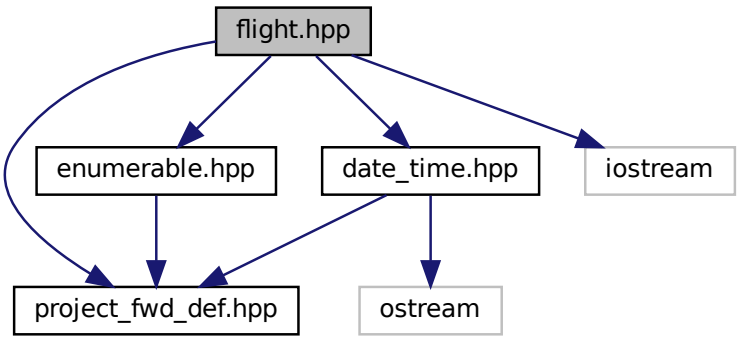
Reference to the output stream.

6.10 flight.hpp File Reference

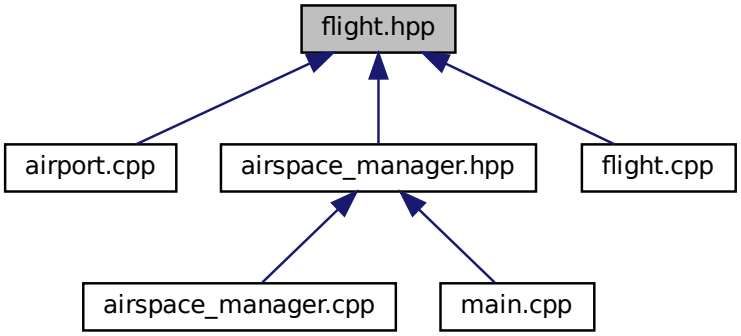
Contains the declaration of the Flight class, representing a flight with associated data and operations.

```
#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

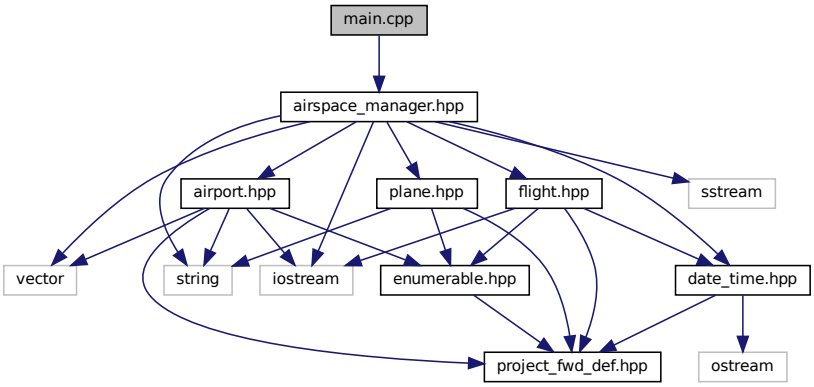
  Represents a flight, including its schedule, route, and assigned plane.

6.10.1 Detailed Description

Contains the declaration of the Flight class, representing a flight with associated data and operations.

6.11 main.cpp File Reference

#include "airspace_manager.hpp"
Include dependency graph for main.cpp:



Functions

- int main ()

6.11.1 Function Documentation

6.11.1.1 main()

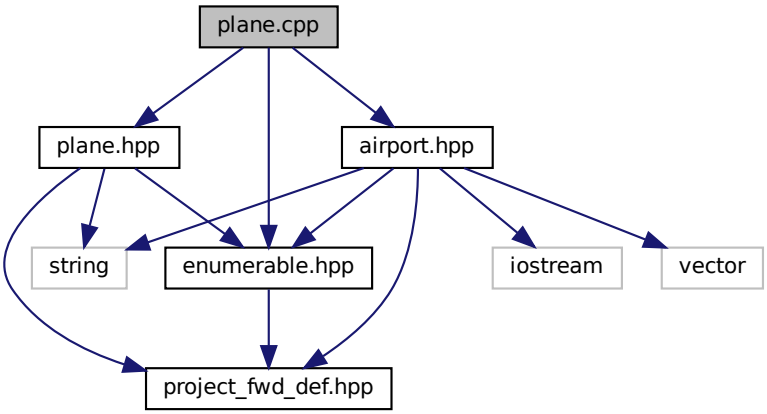
int main ( )

References AirSpaceManager::run().

Here is the call graph for this function:

6.12 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)

6.12.1 Function Documentation

6.12.1.1 operator<<()

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

Parameters

out	The output stream.
plane	The plane object to be output.

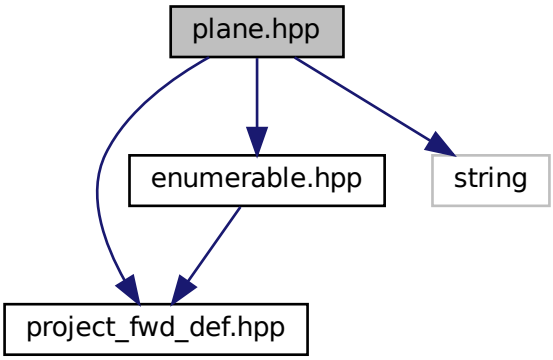
Returns

Reference to the output stream.

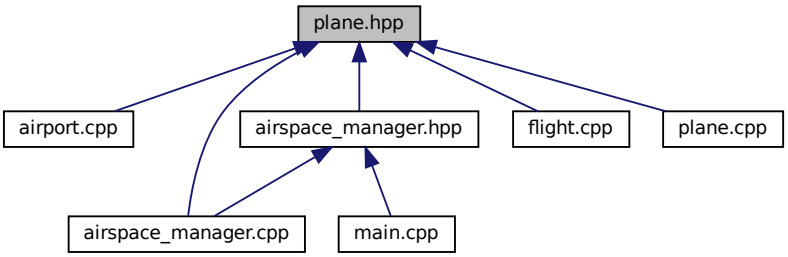
6.13 plane.hpp File Reference

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.

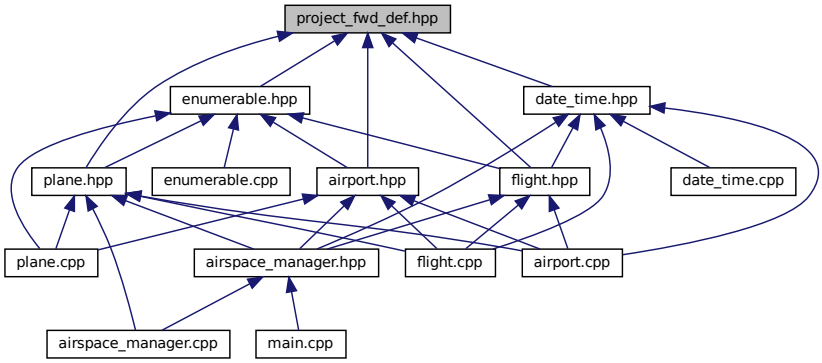
6.13.1 Detailed Description

Contains the declaration of the Plane class and its methods.

6.14 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:



6.14.1 Detailed Description

Contains the declaration of the project_fwd_def class and its methods.

Азбучен указател