

PA03

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	CityNodeV1 Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Constructor & Destructor Documentation	5
3.1.2.1	CityNodeV1(const std::string &name)	5
3.1.2.2	CityNodeV1()	6
3.1.3	Member Data Documentation	6
3.1.3.1	connections	6
3.1.3.2	name	6
3.1.3.3	visited	6
3.2	CityNodeV2 Class Reference	6
3.2.1	Detailed Description	7
3.2.2	Constructor & Destructor Documentation	7
3.2.2.1	CityNodeV2(const std::string &name)	7
3.2.2.2	CityNodeV2()	8
3.2.3	Member Data Documentation	8
3.2.3.1	connections	8
3.2.3.2	name	8

3.2.3.3	tryNext	8
3.2.3.4	visited	8
3.3	ConnectionV1 Struct Reference	8
3.3.1	Detailed Description	8
3.3.2	Member Data Documentation	9
3.3.2.1	cost	9
3.3.2.2	dest	9
3.3.2.3	number	9
3.4	ConnectionV2 Struct Reference	9
3.4.1	Detailed Description	9
3.4.2	Member Data Documentation	9
3.4.2.1	cost	9
3.4.2.2	dest	9
3.4.2.3	number	9
3.5	FlightMapV1 Class Reference	9
3.5.1	Detailed Description	10
3.5.2	Constructor & Destructor Documentation	10
3.5.2.1	FlightMapV1(std::vector< std::string > cities, std::map< std::pair< std::string, std::string >, std::pair< int, int >> flights)	10
3.5.3	Member Function Documentation	10
3.5.3.1	isPath(const std::string &origin, const std::string &dest, std::string &itinerary, std::ofstream &log)	10
3.5.3.2	markVisited(const std::string &city)	11
3.5.3.3	unvisitAll()	11
3.6	FlightMapV2 Class Reference	11
3.6.1	Detailed Description	11
3.6.2	Constructor & Destructor Documentation	11
3.6.2.1	FlightMapV2(std::vector< std::string > cities, std::map< std::pair< std::string, std::string >, std::pair< int, int >> flights)	11
3.6.3	Member Function Documentation	12
3.6.3.1	isPath(const std::string &origin, const std::string &dest, std::string &itinerary, std::ofstream &log)	12
3.6.3.2	markVisited(const std::string &city)	12
3.6.3.3	unvisitAll()	12

4 File Documentation	13
4.1 FlightMap.v1.cpp File Reference	13
4.1.1 Function Documentation	13
4.1.1.1 logStackState(const std::stack< std::string > &stack, std::ofstream &log)	13
4.1.1.2 strInVec(const std::vector< std::string > &vec, const std::string &toFind)	13
4.2 FlightMap.v1.h File Reference	14
4.3 FlightMap.v2.cpp File Reference	15
4.3.1 Function Documentation	15
4.3.1.1 logStackState(const std::stack< std::string > &stack, std::ofstream &log)	15
4.3.1.2 strInVec(const std::vector< std::string > &vec, const std::string &toFind)	15
4.4 FlightMap.v2.cpp File Reference	15
4.4.1 Function Documentation	16
4.4.1.1 logStackState(const std::stack< std::string > &stack, std::ofstream &log)	16
4.4.1.2 strInVec(const std::vector< std::string > &vec, const std::string &toFind)	16
4.5 FlightMap.v2.h File Reference	16
4.6 PA03.cpp File Reference	17
4.6.1 Function Documentation	18
4.6.1.1 loadCities(const std::string &path)	18
4.6.1.2 loadFlights(const std::string &path)	18
4.6.1.3 loadRequests(const std::string &path)	19
4.6.1.4 main(int argc, char **argv)	19
4.6.1.5 splitString(const std::string &toSplit, char delim)	19
4.6.1.6 stringToInt(const std::string &toConvert)	19
Index	21

Chapter 1

Class Index

1.1 Class List

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

CityNodeV1	Represents a v1 node in the city adjacency map	5
CityNodeV2	Represents a v2 node in the city adjacency map	6
ConnectionV1	Represents a v1 connection between cities	8
ConnectionV2	Represents a V2 connection between cities	9
FlightMapV1	Represents a v1 flight map	9
FlightMapV2	Represents a v2 flight map	11

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

FlightMap.v1.cpp	13
FlightMap.v1.h	14
FlightMap.v2.cpp	15
FlightMap.v2.cpp.cpp	15
FlightMap.v2.h	16
PA03.cpp	17

Chapter 3

Class Documentation

3.1 CityNodeV1 Class Reference

Represents a v1 node in the city adjacency map.

```
#include <FlightMap.v1.h>
```

Public Member Functions

- [CityNodeV1](#) (const std::string &[name](#))
Initializes node with the given name.
- [CityNodeV1](#) ()
Initializes node with empty name.

Public Attributes

- std::string [name](#)
- std::vector< [ConnectionV1](#) > [connections](#)
- bool [visited](#)

3.1.1 Detailed Description

Represents a v1 node in the city adjacency map.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 CityNodeV1::CityNodeV1 (const std::string & *name*)

Initializes node with the given name.

Parameters

<i>name</i>	The name of the city.
-------------	-----------------------

3.1.2.2 CityNodeV1::CityNodeV1 ()

Initializes node with empty name.

Note

Required for std::map compatibility.

3.1.3 Member Data Documentation

3.1.3.1 std::vector<ConnectionV1> CityNodeV1::connections

3.1.3.2 std::string CityNodeV1::name

3.1.3.3 bool CityNodeV1::visited

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

- [FlightMap.v1.h](#)
- [FlightMap.v1.cpp](#)
- [FlightMap.v2.cpp.cpp](#)

3.2 CityNodeV2 Class Reference

Represents a v2 node in the city adjacency map.

```
#include <FlightMap.v2.h>
```

Public Member Functions

- [CityNodeV2](#) (const std::string &[name](#))
Initializes node with the given name.
- [CityNodeV2](#) ()
Initializes node with empty name.

Public Attributes

- std::string [name](#)
- std::vector< [ConnectionV2](#) > [connections](#)
- bool [visited](#)
- int [tryNext](#)

3.2.1 Detailed Description

Represents a v2 node in the city adjacency map.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 CityNodeV2::CityNodeV2 (const std::string & *name*)

Initializes node with the given name.

Parameters

<i>name</i>	The name of the city.
-------------	-----------------------

3.2.2.2 CityNodeV2::CityNodeV2 ()

Initializes node with empty name.

Note

Required for std::map compatibility.

3.2.3 Member Data Documentation**3.2.3.1 std::vector<ConnectionV2> CityNodeV2::connections****3.2.3.2 std::string CityNodeV2::name****3.2.3.3 int CityNodeV2::tryNext****3.2.3.4 bool CityNodeV2::visited**

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

- [FlightMap.v2.h](#)
- [FlightMap.v2.cpp](#)

3.3 ConnectionV1 Struct Reference

Represents a v1 connection between cities.

```
#include <FlightMap.v1.h>
```

Public Attributes

- std::string [dest](#)
- int [cost](#)
- int [number](#)

3.3.1 Detailed Description

Represents a v1 connection between cities.

3.3.2 Member Data Documentation

3.3.2.1 `int ConnectionV1::cost`

3.3.2.2 `std::string ConnectionV1::dest`

3.3.2.3 `int ConnectionV1::number`

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

- [FlightMap.v1.h](#)

3.4 ConnectionV2 Struct Reference

Represents a V2 connection between cities.

```
#include <FlightMap.v2.h>
```

Public Attributes

- `std::string dest`
- `int cost`
- `int number`

3.4.1 Detailed Description

Represents a V2 connection between cities.

3.4.2 Member Data Documentation

3.4.2.1 `int ConnectionV2::cost`

3.4.2.2 `std::string ConnectionV2::dest`

3.4.2.3 `int ConnectionV2::number`

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

- [FlightMap.v2.h](#)

3.5 FlightMapV1 Class Reference

Represents a v1 flight map.

```
#include <FlightMap.v1.h>
```

Public Member Functions

- [FlightMapV1](#) (std::vector< std::string > cities, std::map< std::pair< std::string, std::string >, std::pair< int, int >> flights)
Initializes the flight map.
- void [markVisited](#) (const std::string &city)
Mark a city as visited.
- void [unvisitAll](#) ()
Mark all cities as not visited.
- bool [isPath](#) (const std::string &origin, const std::string &dest, std::string &itinerary, std::ofstream &log)
Test whether a sequence of flights exists between two cities.

3.5.1 Detailed Description

Represents a v1 flight map.

3.5.2 Constructor & Destructor Documentation

- 3.5.2.1 [FlightMapV1::FlightMapV1](#) (std::vector< std::string > *cities*, std::map< std::pair< std::string, std::string >, std::pair< int, int >> *flights*)

Initializes the flight map.

Parameters

<i>cities</i>	A list of all cities.
<i>flights</i>	A list of all flights.

3.5.3 Member Function Documentation

- 3.5.3.1 [bool FlightMapV1::isPath](#) (const std::string & *origin*, const std::string & *dest*, std::string & *itinerary*, std::ofstream & *log*)

Test whether a sequence of flights exists between two cities.

Parameters

<i>origin</i>	The name of the origin city.
<i>dest</i>	The name of the destination city.
<i>itinerary</i>	The itenarary output of the path found.
<i>log</i>	The log to write log (debug) output to.

Returns

True if a sequence exists; false otherwise.

3.5.3.2 void FlightMapV1::markVisited (const std::string & city)

Mark a city as visited.

Parameters

<i>city</i>	The name of the city to mark as visited.
-------------	--

Precondition

The given city is registered in the map.

3.5.3.3 void FlightMapV1::unvisitAll ()

Mark all cities as not visited.

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

- [FlightMap.v1.h](#)
- [FlightMap.v1.cpp](#)
- [FlightMap.v2.cpp.cpp](#)

3.6 FlightMapV2 Class Reference

Represents a v2 flight map.

```
#include <FlightMap.v2.h>
```

Public Member Functions

- [FlightMapV2](#) (std::vector< std::string > cities, std::map< std::pair< std::string, std::string >, std::pair< int, int >> flights)
Initializes the flight map.
- void [markVisited](#) (const std::string &city)
Mark a city as visited.
- void [unvisitAll](#) ()
Mark all cities as not visited.
- bool [isPath](#) (const std::string &origin, const std::string &dest, std::string &itinerary, std::ofstream &log)
Test whether a sequence of flights exists between two cities.

3.6.1 Detailed Description

Represents a v2 flight map.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 FlightMapV2::FlightMapV2 (std::vector< std::string > cities, std::map< std::pair< std::string, std::string >, std::pair< int, int >> flights)

Initializes the flight map.

Parameters

<i>cities</i>	A list of all cities.
<i>flights</i>	A list of all flights.

3.6.3 Member Function Documentation

3.6.3.1 `bool FlightMapV2::isPath (const std::string & origin, const std::string & dest, std::string & itinerary, std::ofstream & log)`

Test whether a sequence of flights exists between two cities.

Parameters

<i>origin</i>	The name of the origin city.
<i>dest</i>	The name of the destination city.
<i>itinerary</i>	The itenarary output of the path found.
<i>log</i>	The log to write log (debug) output to.

Returns

True if a sequence exists; false otherwise.

3.6.3.2 `void FlightMapV2::markVisited (const std::string & city)`

Mark a city as visited.

Parameters

<i>city</i>	The name of the city to mark as visited.
-------------	--

Precondition

The given city is registered in the map.

3.6.3.3 `void FlightMapV2::unvisitAll ()`

Mark all cities as not visited.

Note

In V2, also resets tryNext counters.

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

- [FlightMap.v2.h](#)
- [FlightMap.v2.cpp](#)

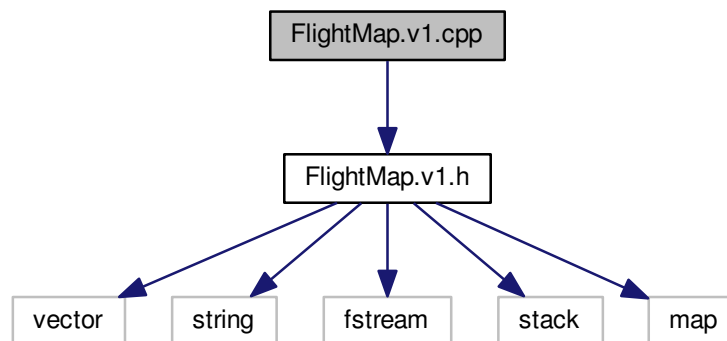
Chapter 4

File Documentation

4.1 FlightMap.v1.cpp File Reference

```
#include "FlightMap.v1.h"
```

Include dependency graph for FlightMap.v1.cpp:



Functions

- bool [strInVec](#) (const std::vector< std::string > &vec, const std::string &toFind)
- void [logStackState](#) (const std::stack< std::string > &stack, std::ofstream &log)

4.1.1 Function Documentation

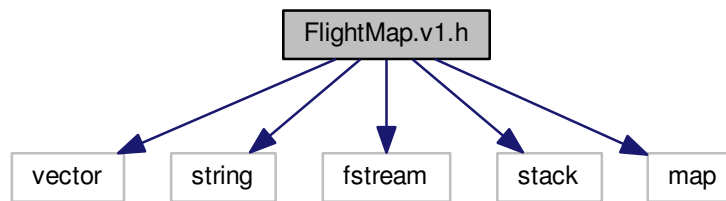
4.1.1.1 void [logStackState](#) (const std::stack< std::string > & *stack*, std::ofstream & *log*)

4.1.1.2 bool [strInVec](#) (const std::vector< std::string > & *vec*, const std::string & *toFind*)

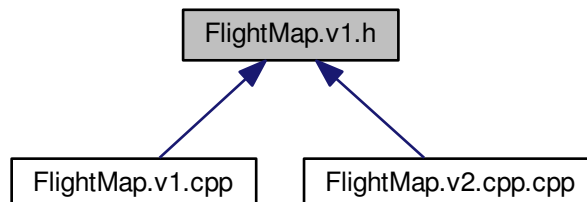
4.2 FlightMap.v1.h File Reference

```
#include <vector>
#include <string>
#include <fstream>
#include <stack>
#include <map>
```

Include dependency graph for FlightMap.v1.h:



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



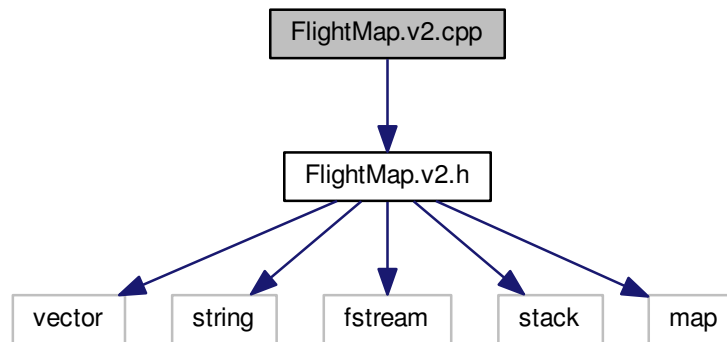
Classes

- struct [ConnectionV1](#)
Represents a v1 connection between cities.
- class [CityNodeV1](#)
Represents a v1 node in the city adjacency map.
- class [FlightMapV1](#)
Represents a v1 flight map.

4.3 FlightMap.v2.cpp File Reference

```
#include "FlightMap.v2.h"
```

Include dependency graph for FlightMap.v2.cpp:



Functions

- bool [strInVec](#) (const std::vector< std::string > &vec, const std::string &toFind)
- void [logStackState](#) (const std::stack< std::string > &stack, std::ofstream &log)

4.3.1 Function Documentation

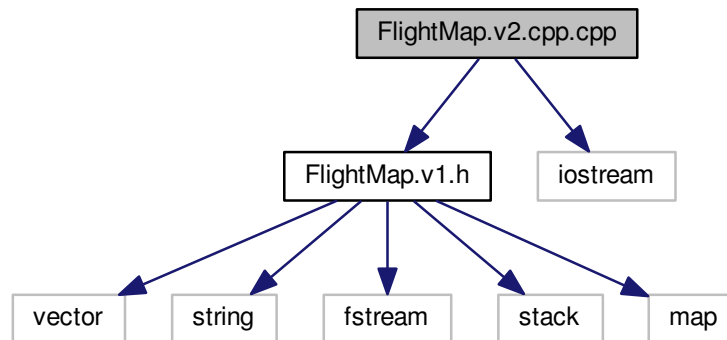
4.3.1.1 void [logStackState](#) (const std::stack< std::string > & *stack*, std::ofstream & *log*)

4.3.1.2 bool [strInVec](#) (const std::vector< std::string > & *vec*, const std::string & *toFind*)

4.4 FlightMap.v2.cpp.cpp File Reference

```
#include "FlightMap.v1.h"  
#include <iostream>
```

Include dependency graph for FlightMap.v2.cpp.cpp:



Functions

- bool [strInVec](#) (const std::vector< std::string > &vec, const std::string &toFind)
- void [logStackState](#) (const std::stack< std::string > &stack, std::ofstream &log)

4.4.1 Function Documentation

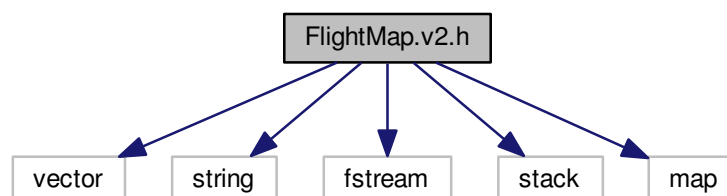
4.4.1.1 void [logStackState](#) (const std::stack< std::string > & *stack*, std::ofstream & *log*)

4.4.1.2 bool [strInVec](#) (const std::vector< std::string > & *vec*, const std::string & *toFind*)

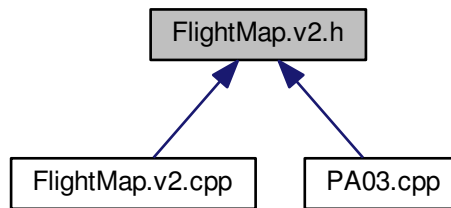
4.5 FlightMap.v2.h File Reference

```
#include <vector>
#include <string>
#include <fstream>
#include <stack>
#include <map>
```

Include dependency graph for FlightMap.v2.h:



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



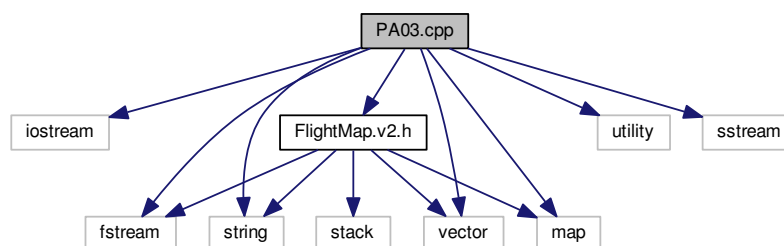
Classes

- struct [ConnectionV2](#)
Represents a V2 connection between cities.
- class [CityNodeV2](#)
Represents a v2 node in the city adjacency map.
- class [FlightMapV2](#)
Represents a v2 flight map.

4.6 PA03.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <map>
#include <utility>
#include <sstream>
#include "FlightMap.v2.h"
```

Include dependency graph for PA03.cpp:



Functions

- `std::vector< std::string > splitString (const std::string &toSplit, char delim)`
Split the given string into parts by the given delimiter.
- `int stringToInt (const std::string &toConvert)`
Convert the given string into an int.
- `std::vector< std::string > loadCities (const std::string &path)`
Load the file containing the list of cities.
- `std::map< std::pair< std::string, std::string >, std::pair< int, int > > loadFlights (const std::string &path)`
Load the file containing the flight paths.
- `std::vector< std::pair< std::string, std::string > > loadRequests (const std::string &path)`
Load the file containing the flight requests.
- `int main (int argc, char **argv)`
Entry point.

4.6.1 Function Documentation

4.6.1.1 `std::vector<std::string> loadCities (const std::string & path)`

Load the file containing the list of cities.

Parameters

<i>path</i>	The path to the file.
-------------	-----------------------

Precondition

The given path points to a readable and correctly formatted file.

Returns

A vector of all the city names read from the file.

4.6.1.2 `std::map<std::pair<std::string, std::string>, std::pair<int, int> > loadFlights (const std::string & path)`

Load the file containing the flight paths.

Parameters

<i>path</i>	The path to the file.
-------------	-----------------------

Precondition

The given path points to a readable and correctly formatted file.

Returns

A map linking string pairs (i.e. ("Origin", "Destination")) to std::pairs of (flight number, flight cost).

4.6.1.3 `std::vector<std::pair<std::string, std::string> > loadRequests (const std::string & path)`

Load the file containing the flight requests.

Parameters

<i>path</i>	The path to the file.
-------------	-----------------------

Precondition

The given path points to a readable and correctly formatted file.

Returns

A vector containing ("Origin", "Destination") pairs.

4.6.1.4 `int main (int argc, char ** argv)`

Entry point.

4.6.1.5 `std::vector<std::string> splitString (const std::string & toSplit, char delim)`

Split the given string into parts by the given delimiter.

Parameters

<i>toSplit</i>	The string to split.
<i>delim</i>	The delimiter.

Returns

A vector containing the parts of the string.

4.6.1.6 `int stringToInt (const std::string & toConvert)`

Convert the given string into an int.

Parameters

<i>toConvert</i>	The string to convert.
------------------	------------------------

Precondition

The given string can be converted into an integer.

Returns

The int value read from the given string.

Index

- CityNodeV1, [5](#)
 - CityNodeV1, [5](#), [6](#)
 - connections, [6](#)
 - name, [6](#)
 - visited, [6](#)
- CityNodeV2, [6](#)
 - CityNodeV2, [7](#), [8](#)
 - connections, [8](#)
 - name, [8](#)
 - tryNext, [8](#)
 - visited, [8](#)
- ConnectionV1, [8](#)
 - cost, [9](#)
 - dest, [9](#)
 - number, [9](#)
- ConnectionV2, [9](#)
 - cost, [9](#)
 - dest, [9](#)
 - number, [9](#)
- connections
 - CityNodeV1, [6](#)
 - CityNodeV2, [8](#)
- cost
 - ConnectionV1, [9](#)
 - ConnectionV2, [9](#)
- dest
 - ConnectionV1, [9](#)
 - ConnectionV2, [9](#)
- FlightMap.v1.cpp, [13](#)
 - logStackState, [13](#)
 - strInVec, [13](#)
- FlightMap.v1.h, [14](#)
- FlightMap.v2.cpp, [15](#)
 - logStackState, [15](#)
 - strInVec, [15](#)
- FlightMap.v2.cpp.cpp, [15](#)
 - logStackState, [16](#)
 - strInVec, [16](#)
- FlightMap.v2.h, [16](#)
- FlightMapV1, [9](#)
 - FlightMapV1, [10](#)
 - isPath, [10](#)
 - markVisited, [10](#)
 - unvisitAll, [11](#)
- FlightMapV2, [11](#)
 - FlightMapV2, [11](#)
 - isPath, [12](#)
 - markVisited, [12](#)
 - unvisitAll, [12](#)
- isPath
 - FlightMapV1, [10](#)
 - FlightMapV2, [12](#)
- loadCities
 - PA03.cpp, [18](#)
- loadFlights
 - PA03.cpp, [18](#)
- loadRequests
 - PA03.cpp, [18](#)
- logStackState
 - FlightMap.v1.cpp, [13](#)
 - FlightMap.v2.cpp, [15](#)
 - FlightMap.v2.cpp.cpp, [16](#)
- main
 - PA03.cpp, [19](#)
- markVisited
 - FlightMapV1, [10](#)
 - FlightMapV2, [12](#)
- name
 - CityNodeV1, [6](#)
 - CityNodeV2, [8](#)
- number
 - ConnectionV1, [9](#)
 - ConnectionV2, [9](#)
- PA03.cpp, [17](#)
 - loadCities, [18](#)
 - loadFlights, [18](#)
 - loadRequests, [18](#)
 - main, [19](#)
 - splitString, [19](#)
 - stringToInt, [19](#)
- splitString
 - PA03.cpp, [19](#)
- strInVec
 - FlightMap.v1.cpp, [13](#)
 - FlightMap.v2.cpp, [15](#)
 - FlightMap.v2.cpp.cpp, [16](#)
- stringToInt
 - PA03.cpp, [19](#)
- tryNext
 - CityNodeV2, [8](#)
- unvisitAll

FlightMapV1, [11](#)
FlightMapV2, [12](#)

visited

CityNodeV1, [6](#)
CityNodeV2, [8](#)