

RouteFinder

Generated by Doxygen 1.8.9.1

Sat Jan 10 2015 12:19:31

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	BsfAlg Class Reference	5
3.1.1	Member Function Documentation	5
3.1.1.1	solve	5
3.2	BsfRandAlg Class Reference	6
3.2.1	Member Function Documentation	6
3.2.1.1	solve	6
3.3	Connection Class Reference	6
3.3.1	Constructor & Destructor Documentation	7
3.3.1.1	Connection	7
3.3.1.2	Connection	7
3.3.1.3	Connection	7
3.3.2	Member Function Documentation	7
3.3.2.1	getArrivalTime	7
3.3.2.2	getDepartureTime	7
3.3.2.3	getTripID	7
3.3.2.4	operator=	7
3.3.3	Friends And Related Function Documentation	8
3.3.3.1	operator<<	8
3.4	DataBase Class Reference	8
3.4.1	Detailed Description	8
3.4.2	Member Enumeration Documentation	9
3.4.2.1	LoadMethod	9
3.4.3	Constructor & Destructor Documentation	9
3.4.3.1	DataBase	9
3.4.4	Member Function Documentation	9

3.4.4.1	isValid	9
3.4.5	Member Data Documentation	9
3.4.5.1	routes	9
3.4.5.2	services	9
3.4.5.3	stops	9
3.4.5.4	stopTimes	9
3.4.5.5	trips	9
3.5	DataReader Class Reference	10
3.5.1	Detailed Description	10
3.5.2	Member Function Documentation	10
3.5.2.1	readRoutes	10
3.5.2.2	readServices	10
3.5.2.3	readStops	10
3.5.2.4	readStopTimes	11
3.5.2.5	readTrips	11
3.6	DsfAlg Class Reference	11
3.6.1	Member Function Documentation	12
3.6.1.1	solve	12
3.7	Edge Class Reference	12
3.7.1	Constructor & Destructor Documentation	12
3.7.1.1	Edge	12
3.7.2	Member Function Documentation	13
3.7.2.1	getEndNode	13
3.7.2.2	getID	13
3.7.2.3	getStartNode	13
3.7.2.4	operator!=	13
3.7.2.5	operator<	13
3.7.2.6	operator=	13
3.7.2.7	operator==	14
3.7.3	Friends And Related Function Documentation	14
3.7.3.1	operator<<	14
3.8	edgePointerCompare Struct Reference	14
3.8.1	Detailed Description	14
3.9	GTFSReader Class Reference	15
3.9.1	Detailed Description	15
3.9.2	Member Function Documentation	15
3.9.2.1	getRoutes	15
3.9.2.2	getServices	15
3.9.2.3	getStops	15
3.9.2.4	getStopTimes	15

3.9.2.5	getTrips	15
3.9.2.6	readGTFS	16
3.9.3	Friends And Related Function Documentation	17
3.9.3.1	operator<<	17
3.10	Network Class Reference	17
3.10.1	Constructor & Destructor Documentation	18
3.10.1.1	Network	18
3.10.1.2	~Network	18
3.10.1.3	Network	18
3.10.2	Member Function Documentation	18
3.10.2.1	findRouteBetween	18
3.10.2.2	getAllEdges	18
3.10.2.3	getAllNodes	18
3.10.2.4	getEdge	18
3.10.2.5	getEdgesForNode	19
3.10.2.6	getNode	19
3.10.2.7	getNodeCloseToPos	19
3.10.2.8	isEdgeBetween	19
3.10.2.9	setSolver	19
3.10.3	Friends And Related Function Documentation	20
3.10.3.1	operator<<	20
3.10.4	Member Data Documentation	20
3.10.4.1	incidenceMatrix	20
3.11	Node Class Reference	20
3.11.1	Constructor & Destructor Documentation	21
3.11.1.1	Node	21
3.11.2	Member Function Documentation	22
3.11.2.1	getID	22
3.11.2.2	getLatitude	22
3.11.2.3	getLongitude	22
3.11.2.4	getName	22
3.11.2.5	operator!=	22
3.11.2.6	operator<	22
3.11.2.7	operator=	23
3.11.2.8	operator==	23
3.11.3	Friends And Related Function Documentation	23
3.11.3.1	operator<<	23
3.12	nodePointerCompare Struct Reference	23
3.13	Route Class Reference	24
3.13.1	Detailed Description	24

3.13.2	Constructor & Destructor Documentation	24
3.13.2.1	Route	24
3.13.3	Member Function Documentation	24
3.13.3.1	addEdge	24
3.13.3.2	begin	25
3.13.3.3	end	25
3.13.3.4	getChangeNumber	25
3.13.3.5	getConnectionsSequence	25
3.13.3.6	getEndNode	25
3.13.3.7	getLength	25
3.13.3.8	getStartNode	26
3.13.3.9	getWeight	26
3.13.3.10	isConnectionBetween	26
3.13.3.11	isEdgeIn	26
3.13.3.12	isNodeIn	26
3.13.3.13	printRoute	26
3.13.3.14	switchEdge	27
3.13.3.15	switchRoute	27
3.13.3.16	validate	27
3.13.4	Friends And Related Function Documentation	27
3.13.4.1	operator<<	27
3.14	RouteData Class Reference	28
3.14.1	Detailed Description	28
3.14.2	Constructor & Destructor Documentation	28
3.14.2.1	RouteData	28
3.14.2.2	RouteData	28
3.14.2.3	RouteData	28
3.14.2.4	~RouteData	29
3.14.3	Member Function Documentation	29
3.14.3.1	getId	29
3.14.3.2	getName	29
3.14.3.3	operator=	29
3.14.3.4	operator==	29
3.14.4	Friends And Related Function Documentation	29
3.14.4.1	operator<<	29
3.15	ServiceData Class Reference	30
3.15.1	Detailed Description	30
3.15.2	Constructor & Destructor Documentation	30
3.15.2.1	ServiceData	30
3.15.2.2	ServiceData	30

3.15.2.3	ServiceData	31
3.15.2.4	~ServiceData	31
3.15.3	Member Function Documentation	31
3.15.3.1	getDays	31
3.15.3.2	getId	31
3.15.3.3	getName	31
3.15.3.4	operator=	31
3.15.3.5	operator==	31
3.15.4	Friends And Related Function Documentation	32
3.15.4.1	operator<<	32
3.16	SimAnnealingAlg Class Reference	32
3.16.1	Detailed Description	32
3.16.2	Member Function Documentation	33
3.16.2.1	solve	33
3.17	Solver Class Reference	33
3.17.1	Detailed Description	33
3.17.2	Member Function Documentation	33
3.17.2.1	solve	33
3.18	StopData Class Reference	34
3.18.1	Detailed Description	34
3.18.2	Constructor & Destructor Documentation	34
3.18.2.1	StopData	34
3.18.2.2	StopData	34
3.18.2.3	StopData	35
3.18.2.4	~StopData	35
3.18.3	Member Function Documentation	35
3.18.3.1	getId	35
3.18.3.2	getLat	35
3.18.3.3	getLng	35
3.18.3.4	getName	35
3.18.3.5	operator=	35
3.18.3.6	operator==	36
3.18.4	Friends And Related Function Documentation	36
3.18.4.1	operator<<	36
3.19	StopTimeData Class Reference	36
3.19.1	Detailed Description	37
3.19.2	Constructor & Destructor Documentation	37
3.19.2.1	StopTimeData	37
3.19.2.2	StopTimeData	37
3.19.2.3	StopTimeData	37

3.19.2.4	~StopTimeData	37
3.19.3	Member Function Documentation	37
3.19.3.1	getId	37
3.19.3.2	getName	38
3.19.3.3	getServiceId	38
3.19.3.4	getStopId	38
3.19.3.5	getStopTime	38
3.19.3.6	getTripId	38
3.19.3.7	operator=	38
3.19.3.8	operator==	38
3.19.4	Friends And Related Function Documentation	39
3.19.4.1	operator<<	39
3.20	Tester Class Reference	40
3.21	Time Class Reference	40
3.21.1	Detailed Description	41
3.21.2	Constructor & Destructor Documentation	41
3.21.2.1	Time	41
3.21.2.2	Time	41
3.21.2.3	Time	41
3.21.2.4	Time	41
3.21.2.5	~Time	41
3.21.3	Member Function Documentation	41
3.21.3.1	operator int	41
3.21.3.2	operator"!="	41
3.21.3.3	operator+	42
3.21.3.4	operator-	42
3.21.3.5	operator<	42
3.21.3.6	operator=	42
3.21.3.7	operator==	42
3.21.3.8	operator>	43
3.21.4	Friends And Related Function Documentation	43
3.21.4.1	operator<<	43
3.22	TripData Class Reference	43
3.22.1	Detailed Description	44
3.22.2	Constructor & Destructor Documentation	44
3.22.2.1	TripData	44
3.22.2.2	TripData	44
3.22.2.3	TripData	44
3.22.2.4	~TripData	44
3.22.3	Member Function Documentation	44

3.22.3.1	getId	44
3.22.3.2	getName	44
3.22.3.3	getRouteId	45
3.22.3.4	getStopSec	45
3.22.3.5	operator=	45
3.22.3.6	operator==	45
3.22.4	Friends And Related Function Documentation	45
3.22.4.1	operator<<	45
Index		47

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Connection	6
DataBase	8
DataReader	10
Edge	12
edgePointerCompare	14
GTFSReader	15
Network	17
Node	20
nodePointerCompare	23
Route	24
RouteData	28
ServiceData	30
Solver	33
BsfAlg	5
BsfRandAlg	6
DsfAlg	11
SimAnnealingAlg	32
StopData	34
StopTimeData	36
Tester	40
Time	40
TripData	43

Chapter 2

Class Index

2.1 Class List

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

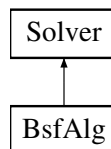
BsfAlg	5
BsfRandAlg	6
Connection	6
DataBase	8
DataReader	10
DsfAlg	11
Edge	12
edgePointerCompare	14
GTFSReader	15
Network	17
Node	20
nodePointerCompare	23
Route	24
RouteData	28
ServiceData	30
SimAnnealingAlg	32
Solver	33
StopData	34
StopTimeData	36
Tester	40
Time	40
TripData	43

Chapter 3

Class Documentation

3.1 BsfAlg Class Reference

Inheritance diagram for BsfAlg:



Public Member Functions

- virtual [Route](#) * [solve](#) (const [Network](#) **n*, [Node](#) **start*, [Node](#) **end*, [Time](#) *t*)
- virtual const std::string & [getName](#) () const

3.1.1 Member Function Documentation

3.1.1.1 [Route](#) * [BsfAlg::solve](#) (const [Network](#) * *n*, [Node](#) * *start*, [Node](#) * *end*, [Time](#) *t*) [virtual]

Method used in [Network](#) class for finding best connection between points. This method need to be implemented in any class inheriting from [Solver](#) class.

Parameters

<i>n</i>	Pointer to Network in which Route is being searched for.
----------	--

Returns

Pointer to found [Route](#), NULL if no route can be found.

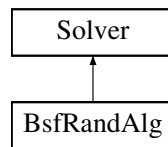
Implements [Solver](#).

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

- /home/vka/Programming/C/workspace/RouteFinder/src/algorithm/BSF.h
- /home/vka/Programming/C/workspace/RouteFinder/src/algorithm/BSF.cpp

3.2 BsfRandAlg Class Reference

Inheritance diagram for BsfRandAlg:



Public Member Functions

- virtual [Route](#) * [solve](#) (const [Network](#) *n, [Node](#) *start, [Node](#) *end, [Time](#) t)
- virtual const std::string & [getName](#) () const

3.2.1 Member Function Documentation

3.2.1.1 [Route](#) * [BsfRandAlg::solve](#) (const [Network](#) * n, [Node](#) * start, [Node](#) * end, [Time](#) t) [virtual]

Method used in [Network](#) class for finding best connection between points. This method need to be implemented in any class inheriting from [Solver](#) class.

Parameters

<i>n</i>	Pointer to Network in which Route is being searched for.
----------	--

Returns

Pointer to found [Route](#), NULL if no route can be found.

Implements [Solver](#).

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

- /home/vka/Programming/C/workspace/RouteFinder/src/algorithm/BSFR.h
- /home/vka/Programming/C/workspace/RouteFinder/src/algorithm/BSFR.cpp

3.3 Connection Class Reference

Public Member Functions

- [Connection](#) ()
- [Connection](#) ([Time](#) departureTime, [Time](#) arrivalTime, unsigned int tripld)
- [Connection](#) (const [Connection](#) &con)
- [Connection](#) & [operator=](#) (const [Connection](#) &con)
- [Time](#) [getDepartureTime](#) () const
- [Time](#) [getArrivalTime](#) () const
- unsigned int [getTripID](#) () const

Friends

- std::ostream & [operator<<](#) (std::ostream &output, [Connection](#) con)

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Connection::Connection ()

Default constructor which makes connection with zero id and times

3.3.1.2 Connection::Connection (Time *departureTime*, Time *arrivalTime*, unsigned int *tripId*)

Constructor

Parameters

<i>departureTime</i>	time when bus leaves starting stop
<i>arrivalTime</i>	time when bus goes on ending stop
<i>tripId</i>	trip of bus

3.3.1.3 Connection::Connection (const Connection & *con*)

Constructor which copies object

Parameters

<i>con</i>	object to copy
------------	----------------

3.3.2 Member Function Documentation

3.3.2.1 Time Connection::getArrivalTime () const

get arrival time

Returns

time when bus goes on ending stop

3.3.2.2 Time Connection::getDepartureTime () const

get departure time

Returns

time when bus leaves starting stop

3.3.2.3 unsigned int Connection::getTripID () const

get id of trip which makes connection

Returns

id of trip

3.3.2.4 Connection & Connection::operator= (const Connection & *con*)

Assign one object to another

Parameters

<i>con</i>	object to copy
------------	----------------

Returns

object which was copied in

3.3.3 Friends And Related Function Documentation

3.3.3.1 `std::ostream& operator<< (std::ostream & output, Connection con)` [*friend*]

print connection to stream

Parameters

<i>output</i>	stream where information will be print to object to printing stream where object were printed
---------------	---

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

- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Connection.h
- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Connection.cpp

3.4 DataBase Class Reference

```
#include <DataBase.h>
```

Public Types

- enum [LoadMethod](#) {
JSON = 0, **GTFS** = 1, **MULTJSON** = 2, **SAVEDDB** = 3,
EMPTY = 4 }

Public Member Functions

- [DataBase](#) ([DataBase::LoadMethod](#) method, std::string path)
- bool [isValid](#) ()
- void **saveToFile** (const std::string p) const

Public Attributes

- std::vector< [RouteData](#) > [routes](#)
- std::vector< [TripData](#) > [trips](#)
- std::vector< [StopData](#) > [stops](#)
- std::vector< [StopTimeData](#) > [stopTimes](#)
- std::vector< [ServiceData](#) > [services](#)
- std::vector< std::vector< std::vector< [Time](#) > > > [timeTable](#)

3.4.1 Detailed Description

Database class, used to loading from files and then being converted into [Network](#) object. Provides input from gtfs or json formats.

3.4.2 Member Enumeration Documentation

3.4.2.1 enum DataBase::LoadMethod

Enum defining loading method being used.

3.4.3 Constructor & Destructor Documentation

3.4.3.1 DataBase::DataBase (DataBase::LoadMethod *method*, std::string *path*)

Constructor.

Parameters

<i>method</i>	Defines load method being used.
<i>path</i>	Provides path to files being loaded. If one file is being used it needs to get path to file, otherwise - to directory containing files.

3.4.4 Member Function Documentation

3.4.4.1 bool DataBase::isValid ()

Method checking validity of loaded database.

Returns

true if all vectors got populated with data, false otherwise.

3.4.5 Member Data Documentation

3.4.5.1 std::vector<RouteData> DataBase::routes

std::vector object containing loaded [RouteData](#).

3.4.5.2 std::vector<ServiceData> DataBase::services

std::vector object containing loaded [ServiceData](#).

3.4.5.3 std::vector<StopData> DataBase::stops

std::vector object containing loaded [StopData](#).

3.4.5.4 std::vector<StopTimeData> DataBase::stopTimes

std::vector object containing loaded [StopTimeData](#).

3.4.5.5 std::vector<TripData> DataBase::trips

std::vector object containing loaded [TripData](#).

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

- /home/vka/Programming/C/workspace/RouteFinder/src/db/DataBase.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/DataBase.cpp

3.5 DataReader Class Reference

```
#include <DataReader.h>
```

Static Public Member Functions

- static std::vector< [RouteData](#) > [readRoutes](#) (std::string filename, bool oneFile=true)
- static std::vector< [StopData](#) > [readStops](#) (std::string filename, bool oneFile=true)
- static std::vector< [TripData](#) > [readTrips](#) (std::string filename, bool oneFile=true)
- static std::vector< [StopTimeData](#) > [readStopTimes](#) (std::string filename, bool oneFile=true)
- static std::vector< [ServiceData](#) > [readServices](#) (std::string filename, bool oneFile=true)

3.5.1 Detailed Description

[DataReader](#) reads data from one or multiple json files. If oneFile is set, functions assume that whole database is contained in given file, otherwise, assume given file contains only necessary data and so std::vector sth = root; (no root['sth'] is needed.)

3.5.2 Member Function Documentation

3.5.2.1 std::vector< [RouteData](#) > [DataReader::readRoutes](#) (std::string *filename*, bool *oneFile* = true) [static]

Function loading routes from given file.

Parameters

<i>filename</i>	path to file.
<i>oneFile</i>	Defines if all data is being loaded from oneFile.

Returns

vector containing [RouteData](#).

3.5.2.2 std::vector< [ServiceData](#) > [DataReader::readServices](#) (std::string *filename*, bool *oneFile* = true) [static]

Function loading services from given file.

Parameters

<i>filename</i>	path to file.
<i>oneFile</i>	Defines if all data is being loaded from oneFile.

Returns

vector containing [ServiceData](#).

3.5.2.3 std::vector< [StopData](#) > [DataReader::readStops](#) (std::string *filename*, bool *oneFile* = true) [static]

Function loading stops from given file.

Parameters

<i>filename</i>	path to file.
<i>oneFile</i>	Defines if all data is being loaded from oneFile.

Returns

vector containing [StopData](#).

3.5.2.4 `std::vector< StopTimeData > DataReader::readStopTimes (std::string filename, bool oneFile = true)`
`[static]`

Function loading stop times from given file.

Parameters

<i>filename</i>	path to file.
<i>oneFile</i>	Defines if all data is being loaded from oneFile.

Returns

vector containing [StopTimeData](#).

3.5.2.5 `std::vector< TripData > DataReader::readTrips (std::string filename, bool oneFile = true)` `[static]`

Function loading trips from given file.

Parameters

<i>filename</i>	path to file.
<i>oneFile</i>	Defines if all data is being loaded from oneFile.

Returns

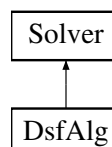
vector containing [TripData](#).

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

- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/DataReader.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/DataReader.cpp

3.6 DsfAlg Class Reference

Inheritance diagram for DsfAlg:



Public Member Functions

- virtual [Route](#) * [solve](#) (const [Network](#) *n, [Node](#) *start, [Node](#) *end, [Time](#) t)
- virtual const std::string & [getName](#) () const

3.6.1 Member Function Documentation

3.6.1.1 `Route * DsfAlg::solve (const Network * n, Node * start, Node * end, Time t)` [virtual]

Method used in [Network](#) class for finding best connection between points. This method need to be implemented in any class inheriting from [Solver](#) class.

Parameters

<code>n</code>	Pointer to Network in which Route is being searched for.
----------------	--

Returns

Pointer to found [Route](#), NULL if no route can be found.

Implements [Solver](#).

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

- `/home/vka/Programming/C/workspace/RouteFinder/src/algorithm/DSF.h`
- `/home/vka/Programming/C/workspace/RouteFinder/src/algorithm/DSF.cpp`

3.7 Edge Class Reference

Public Member Functions

- [Edge](#) (unsigned int id, [Node](#) *start, [Node](#) *end)
- **Edge** (const [Edge](#) &e)
- unsigned int [getID](#) () const
- [Node](#) * [getStartNode](#) () const
- [Node](#) * [getEndNode](#) () const
- bool [operator==](#) (const [Edge](#) &e) const
- bool [operator!=](#) (const [Edge](#) &e) const
- bool [operator<](#) (const [Edge](#) &e) const
- [Edge](#) & [operator=](#) (const [Edge](#) &e)
- void **addConnection** ([Time](#) departureTime, [Time](#) arrivalTime, unsigned int tripId)
- [Time](#) [getNextTime](#) ([Time](#) t) const

Public Attributes

- `std::vector< Connection > connections`

Friends

- `std::ostream & operator<< (std::ostream &s, const Edge &e)`

3.7.1 Constructor & Destructor Documentation

3.7.1.1 `Edge::Edge (unsigned int id, Node * start, Node * end)`

Constructor of [Edge](#) object.

Parameters

<i>id</i>	Identificator of object.
<i>start</i>	Pointer to Node assigned as starting position.
<i>end</i>	Pointer to Node assigned as ending position.
<i>weight</i>	Given weight.
<i>type</i>	Enumerated value describing type of route.

3.7.2 Member Function Documentation

3.7.2.1 `Node * Edge::getEndNode () const`

Returns

Returns pointer to ending [Node](#).

3.7.2.2 `unsigned int Edge::getID () const`

Returns

Returns id of itself.

3.7.2.3 `Node * Edge::getStartNode () const`

Returns

Returns pointer to starting [Node](#).

3.7.2.4 `bool Edge::operator!= (const Edge & e) const`

Compares itself id with given edge id.

Parameters

<i>e</i>	Given Edge .
----------	------------------------------

Returns

True if ids are not equal, false otherwise.

3.7.2.5 `bool Edge::operator< (const Edge & e) const`

Compares itself id with given edge id. Method used in [Network](#) class.

Parameters

<i>e</i>	Given Edge .
----------	------------------------------

Returns

True if this->id is smaller than e.id, false otherwise.

3.7.2.6 `Edge & Edge::operator= (const Edge & e)`

Assign operator.

Parameters

<i>e</i>	Given Edge .
----------	------------------------------

Returns

Reference to itself.

3.7.2.7 bool Edge::operator==(const Edge & e) const

Compares itself id with given edge id.

Parameters

<i>e</i>	Given Edge .
----------	------------------------------

Returns

True if ids are equal, false otherwise.

3.7.3 Friends And Related Function Documentation**3.7.3.1 std::ostream& operator<< (std::ostream & s, const Edge & e) [friend]**

Operator used for console debug purposes.

Parameters

<i>s</i>	Stream which is used for output.
<i>e</i>	Edge on which operator is called.

Returns

Given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Edge.h
- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Edge.cpp

3.8 edgePointerCompare Struct Reference

```
#include <Network.h>
```

Public Member Functions

- bool **operator()** ([Edge](#) *e1, [Edge](#) *e2) const

3.8.1 Detailed Description

main class, contains information about nodes and edges between them. Should be created from file containing data in GTFS or other format. loadFromFile method should load "db/db.ext" file and save it to inner variables.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Network.h

3.9 GTFSReader Class Reference

```
#include <GTFSReader.h>
```

Public Member Functions

- void [readGTFS](#) (std::string filename)
- std::vector< [RouteData](#) > [getRoutes](#) ()
- std::vector< [StopData](#) > [getStops](#) ()
- std::vector< [TripData](#) > [getTrips](#) ()
- std::vector< [StopTimeData](#) > [getStopTimes](#) ()
- std::vector< [ServiceData](#) > [getServices](#) ()

Friends

- std::ostream & [operator<<](#) (std::ostream &stream, const [GTFSReader](#) &reader)

3.9.1 Detailed Description

[GTFSReader](#) reads data from GTFS format zip archive. It is compatible with [DataReader](#) class.

3.9.2 Member Function Documentation

3.9.2.1 std::vector< [RouteData](#) > GTFSReader::getRoutes ()

Returns

vector containing [RouteData](#).

3.9.2.2 std::vector< [ServiceData](#) > GTFSReader::getServices ()

Returns

vector containing [ServiceData](#).

3.9.2.3 std::vector< [StopData](#) > GTFSReader::getStops ()

Returns

vector containing [StopData](#).

3.9.2.4 std::vector< [StopTimeData](#) > GTFSReader::getStopTimes ()

Returns

vector containing [StopTimeData](#).

3.9.2.5 std::vector< [TripData](#) > GTFSReader::getTrips ()

Returns

vector containing [TripData](#).

3.9.2.6 void GTFSReader::readGTFS (std::string *filename*)

Unpacks gtfs archive, creates network, deletes created in progress files.

Parameters

<i>filename</i>	Path to gtfs file.
-----------------	--------------------

3.9.3 Friends And Related Function Documentation

3.9.3.1 `std::ostream& operator<< (std::ostream & stream, const GTFSReader & reader)` [friend]

Helper output function.

Parameters

<i>stream</i>	Stream
<i>reader</i>	Outputed object.

Returns

Reference to given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/GTFSReader.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/GTFSReader.cpp

3.10 Network Class Reference

Public Member Functions

- [Network](#) ()
- [~Network](#) ()
- [Network](#) ([DataBase](#) &dataB)
- void [setSolver](#) ([Solver](#) *s)
- [Route](#) * [findRouteBetween](#) ([Node](#) *start, [Node](#) *end, [Time](#) t)
- std::list< [Node](#) * > [getAllNodes](#) () const
- std::list< [Edge](#) * > [getAllEdges](#) () const
- bool [isEdgeBetween](#) (const [Node](#) *start, const [Node](#) *end) const
- [Node](#) * [getNode](#) (unsigned int id) const
- [Edge](#) * [getEdge](#) (unsigned int id) const
- std::list< [Edge](#) * > [getEdgesForNode](#) (const [Node](#) *n) const
- [Node](#) * [getNodeCloseToPos](#) (double latitude, double longitude) const
- unsigned int [calculateEdgeld](#) (unsigned int startId, unsigned int endId) const
- bool [validate](#) ()

Static Public Member Functions

- static [Network](#) * [generateRandomNetwork](#) (unsigned width, unsigned height, long seed=0, double probability=1)

Public Attributes

- bool ** [incidenceMatrix](#)

Friends

- `std::ostream & operator<< (std::ostream &s, const Network &n)`

3.10.1 Constructor & Destructor Documentation

3.10.1.1 `Network::Network ()`

[Network](#) object constructor. If this constructor is called, `loadFromFile` method need to be called after.

3.10.1.2 `Network::~~Network ()`

[Network](#) object constructor in which `Network::loadFromFile()` method is being called.

Parameters

<i>f</i>	Name of file from which database is loaded. Destructs all objects in Network and itself.
----------	--

3.10.1.3 `Network::Network (DataBase & dataB)`

Creates [Network](#) from database

3.10.2 Member Function Documentation

3.10.2.1 `Route * Network::findRouteBetween (Node * start, Node * end, Time t)`

Searches for [Route](#) between two given points.

Parameters

<i>start</i>	Start Node .
<i>end</i>	End Node .

Returns

Pointer to [Route](#) between given nodes, NULL if no route can be found.

3.10.2.2 `std::list< Edge * > Network::getAllEdges () const`

For debug.

Returns

Returns some kind of stl container in which all edges are stored.

3.10.2.3 `std::list< Node * > Network::getAllNodes () const`

This function is necessary for GUI.

Returns

Returns some kind of stl container in which all nodes are stored in alphabetical order.

3.10.2.4 `Edge * Network::getEdge (unsigned int id) const`

Parameters

<i>id</i>	Id of desired Edge .
-----------	--------------------------------------

Returns

Pointer to desired [Edge](#) if exists in graph, NULL otherwise.

3.10.2.5 `std::list< Edge * > Network::getEdgesForNode (const Node * n) const`

Parameters

<i>n</i>	Pointer to given Node .
----------	---

Returns

std::list of all Edges starting in given [Node](#).

3.10.2.6 `Node * Network::getNode (unsigned int id) const`

Parameters

<i>id</i>	Id of desired Node .
-----------	--------------------------------------

Returns

Pointer to desired [Node](#) if exists in graph, NULL otherwise.

3.10.2.7 `Node * Network::getNodeCloseToPos (double latitude, double longitude) const`

Parameters

<i>latitude</i>	Given latitude
<i>longitude</i>	Given longitude.

Returns

Returns pointer to [Node](#) being closest to given geographic position.

3.10.2.8 `bool Network::isEdgeBetween (const Node * start, const Node * end) const`

Parameters

<i>start</i>	Pointer to starting Node .
<i>end</i>	Pointer to ending Node .

Returns

True if [Edge](#) from start to end exists in graph, false otherwise.

3.10.2.9 `void Network::setSolver (Solver * s)`

Set solved used in [Network::findRouteBetween\(\)](#) method.

Parameters

<i>s</i>	Pointer to Solver being used.
----------	---

3.10.3 Friends And Related Function Documentation

3.10.3.1 `std::ostream& operator<< (std::ostream & s, const Network & n)` [*friend*]

Used of debug in console purposes.

Parameters

<i>s</i>	Stream used for output.
<i>n</i>	Reference to Network being printed.

Returns

Given stream.

3.10.4 Member Data Documentation

3.10.4.1 `bool** Network::incidenceMatrix`

two dimensional bool array containing data about connections for matrix[i][j], true means there is connection between node.id == i to node.id ==j.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Network.h
- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Network.cpp

3.11 Node Class Reference

Public Member Functions

- [Node](#) (unsigned int id, std::string name, double lon, double lat)
- double [getLongitude](#) () const
- double [getLatitude](#) () const
- unsigned int [getID](#) () const
- std::string [getName](#) () const
- std::list< [Edge](#) * > [getEdges](#) () const
- bool [addEdge](#) ([Edge](#) *e)
- bool [operator==](#) (const [Node](#) &n) const
- bool [operator!=](#) (const [Node](#) &n) const
- bool [operator<](#) (const [Node](#) &n) const
- [Node](#) & [operator=](#) (const [Node](#) &n)

Friends

- std::ostream & [operator<<](#) (std::ostream &s, const [Node](#) &n)

3.11.1 Constructor & Destructor Documentation

3.11.1.1 Node::Node (unsigned int *id*, std::string *name*, double *lon*, double *lat*)

Constructs new [Node](#) object.

Parameters

<i>id</i>	Id of an node.
<i>name</i>	Name of node (stop).
<i>lon</i>	Longitude coord.
<i>lat</i>	Latitude coord.

3.11.2 Member Function Documentation

3.11.2.1 unsigned int Node::getID () const

Returns

Returns id of itself.

3.11.2.2 double Node::getLatitude () const

Returns

Returns latitude as double.

3.11.2.3 double Node::getLongitude () const

Returns

Returns longitude as double.

3.11.2.4 std::string Node::getName () const

Returns

Returns string containing name.

3.11.2.5 bool Node::operator!= (const Node & n) const

Operator compares id of this and given node.

Parameters

<i>n</i>	Node which is being compared.
----------	---

Returns

False if ids are equal, true otherwise.

3.11.2.6 bool Node::operator< (const Node & n) const

Operator used in sets in [Network](#) class. Compares ids.

Parameters

<i>n</i>	Node which is being compared.
----------	---

Returns

True if this->id is smaller than n.id, false otherwise.

3.11.2.7 Node & Node::operator= (const Node & *n*)

Copies params from given [Node](#) to itself.

Parameters

<i>n</i>	Node which is being copied.
----------	---

Returns

Reference to itself.

3.11.2.8 bool Node::operator== (const Node & *n*) const

Operator compares id of this and given node.

Parameters

<i>n</i>	Node which is being compared.
----------	---

Returns

True if ids are equal, false otherwise.

3.11.3 Friends And Related Function Documentation

3.11.3.1 std::ostream& operator<< (std::ostream & *s*, const Node & *n*) [friend]

Operator used for console debug purposes.

Parameters

<i>s</i>	Stream which is used for output.
<i>n</i>	Node on which operator is called.

Returns

Given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Node.h
- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Node.cpp

3.12 nodePointerCompare Struct Reference

Public Member Functions

- bool **operator()** ([Node](#) *n1, [Node](#) *n2) const

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

- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Network.h

3.13 Route Class Reference

```
#include <Route.h>
```

Public Member Functions

- [Route](#) ()
- **Route** (const [Route](#) &r)
- unsigned int [getLength](#) () const
- unsigned int [getWeight](#) ([Time](#) t) const
- bool [validate](#) () const
- bool [addEdge](#) (const [Edge](#) *e)
- bool [switchEdge](#) (const [Edge](#) *e)
- bool [switchRoute](#) ([Route](#) &r)
- const [Node](#) * [getStartNode](#) () const
- const [Node](#) * [getEndNode](#) () const
- bool [isNodeIn](#) (const [Node](#) *n) const
- bool [isEdgeIn](#) (const [Edge](#) *e) const
- bool [isConnectionBetween](#) (const [Node](#) *start, const [Node](#) *end) const
- std::list< const [Edge](#) * >::iterator [begin](#) ()
- std::list< const [Edge](#) * >::iterator [end](#) ()
- std::vector< [Connection](#) > [getConnectionsSequence](#) ([Time](#) t) const
- unsigned int [getChangeNumber](#) ([Time](#) t) const

Static Public Member Functions

- static void [printRoute](#) (std::ostream &output, [Route](#) *r, [Time](#) t)

Friends

- std::ostream & [operator<<](#) (std::ostream &s, [Route](#) &r)

3.13.1 Detailed Description

contains information about route between two points Should be used in solver class.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 [Route::Route](#) ()

Constructs object. No data is necessary.

3.13.3 Member Function Documentation

3.13.3.1 [bool Route::addEdge](#) (const [Edge](#) * e)

Add one [Edge](#) to the end of [Route](#). Given edge must start in [Node](#), in which current route ends.

Parameters

<i>e</i>	Pointer to given edge.
----------	------------------------

Returns

True if edge could be connected to route, false otherwise.

3.13.3.2 `std::list< const Edge * >::iterator Route::begin ()`

Returns

Returns iterator to the beginning of route.

3.13.3.3 `std::list< const Edge * >::iterator Route::end ()`

Returns

Returns iterator to point after last [Edge](#) in route.

3.13.3.4 `unsigned int Route::getChangeNumber (Time t) const`

calculate how many times we have to change trip while traveling this route

Parameters

<i>t</i>	time when route starts
----------	------------------------

Returns

number of changes

3.13.3.5 `std::vector< Connection > Route::getConnectionsSequence (Time t) const`

Returns connection sequence on witch `getWeight(Time)` calculate result

Parameters

<i>t</i>	Time when calculating starts from
----------	---

Returns

vector with found connections

3.13.3.6 `const Node * Route::getEndNode () const`

Returns

pointer to [Node](#) on the end of [Route](#).

3.13.3.7 `unsigned int Route::getLength () const`

Returns

Returns length - number of [Edge](#) objects.

3.13.3.8 `const Node * Route::getStartNode () const`

Returns

Returns pointer to [Node](#) on the begining of [Route](#).

3.13.3.9 `unsigned int Route::getWeight (Time t) const`

Returns

Returns sum of weights of [Edge](#) objects.

3.13.3.10 `bool Route::isConnectionBetween (const Node * start, const Node * end) const`

Checks for subroute between given nodes.

Parameters

<i>start</i>	Pointer to Node of start.
<i>end</i>	Pinter to Node of end.

Returns

True if there is subroute from start to end in route, false otherwise.

3.13.3.11 `bool Route::isEdgeln (const Edge * e) const`

Parameters

<i>e</i>	Pointer to Edge which is being searched for.
----------	--

Returns

True if [Edge](#) is included in route, false otherwise.

3.13.3.12 `bool Route::isNodeIn (const Node * n) const`

Parameters

<i>n</i>	Pointer to Node which is being checked.
----------	---

Returns

True if given node is currently in route. False otherwise.

3.13.3.13 `void Route::printRoute (std::ostream & output, Route * r, Time t) [static]`

prints route in "sophisticated" way

Parameters

<i>output</i>	stream to print route in
<i>r</i>	route to print
<i>t</i>	time when route starts

3.13.3.14 bool Route::switchEdge (const Edge * e)

Switches given edge with one included in path,

Parameters

<i>e</i>	Pointer to Edge . Its start Node and end Node must be same as start and end nodes of edge included in route.
----------	--

Returns

True if switch was successful, false otherwise.

3.13.3.15 bool Route::switchRoute (Route & r)

Switches part of [Route](#) with given route.

Parameters

<i>r</i>	Reference to subroute which needs to be inserted into object. It must to be correct (validate method is being called), and start and end of subroute must have corresponding values as start and end of some subroute inside current object. Length of switched subroutes do not need to be equal.
----------	--

Returns

True if switch was successful, false otherwise.

3.13.3.16 bool Route::validate () const

Checks if [Route](#) does not contain any loops and if all edges are connected. I.e. A->B and then B->C is ok, but A->B and C->D is wrong.

Returns

True if test is passed, false otherwise.

3.13.4 Friends And Related Function Documentation

3.13.4.1 std::ostream& operator<< (std::ostream & s, Route & r) [friend]

Used of debug in console purposes.

Parameters

<i>s</i>	Stream used for output.
----------	-------------------------

<i>r</i>	Reference to Route being printed.
----------	---

Returns

Given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Route.h
- /home/vka/Programming/C/workspace/RouteFinder/src/graph/Route.cpp

3.14 RouteData Class Reference

```
#include <RouteData.h>
```

Public Member Functions

- [RouteData](#) ()
- [RouteData](#) (const [RouteData](#) &src)
- [RouteData](#) (std::string name, unsigned int id)
- [~RouteData](#) ()
- [RouteData](#) & [operator=](#) (const [RouteData](#) &src)
- bool [operator==](#) (const [RouteData](#) &src)
- std::string [getName](#) () const
- unsigned int [getId](#) () const

Friends

- std::ostream & [operator<<](#) (std::ostream &output, const [RouteData](#) &src)

3.14.1 Detailed Description

Class containing data about routes when database is being created. Then converted into [Route](#) object.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 [RouteData::RouteData](#) ()

Default constructor.

3.14.2.2 [RouteData::RouteData](#) (const [RouteData](#) & *src*)

Copying constructor.

Parameters

<i>src</i>	Object being copied.
------------	----------------------

3.14.2.3 [RouteData::RouteData](#) (std::string *name*, unsigned int *id*)

Constructor.

Parameters

<i>name</i>	Name of route.
<i>id</i>	id of route.

3.14.2.4 RouteData::~~RouteData ()

Destructor.

3.14.3 Member Function Documentation

3.14.3.1 unsigned int RouteData::getId () const

Returns

return id value.

3.14.3.2 std::string RouteData::getName () const

Returns

Returns name value.

3.14.3.3 RouteData & RouteData::operator= (const RouteData & src)

Assignment operator.

Parameters

<i>src</i>	Reference object being assigned.
------------	----------------------------------

Returns

Reference to self.

3.14.3.4 bool RouteData::operator== (const RouteData & src)

Comparison operator.

Parameters

<i>src</i>	Reference to compared object.
------------	-------------------------------

Returns

True if name and id are equal.

3.14.4 Friends And Related Function Documentation

3.14.4.1 std::ostream& operator<< (std::ostream & output, const RouteData & src) [friend]

Helper output function.

Parameters

<i>output</i>	Stream
<i>src</i>	Outputed object.

Returns

Reference to given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/RouteData.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/RouteData.cpp

3.15 ServiceData Class Reference

```
#include <ServiceData.h>
```

Public Member Functions

- [ServiceData](#) ()
- [ServiceData](#) (const [ServiceData](#) &src)
- [ServiceData](#) (unsigned int id, std::string name, unsigned operationalDays)
- [~ServiceData](#) ()
- [ServiceData](#) & [operator=](#) (const [ServiceData](#) &src)
- bool [operator==](#) (const [ServiceData](#) src)
- unsigned int [getId](#) ()
- std::string [getName](#) ()
- unsigned [getDays](#) ()

Friends

- std::ostream & [operator<<](#) (std::ostream &output, const [ServiceData](#) &src)

3.15.1 Detailed Description

Class containing data about services when database is being created.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 [ServiceData::ServiceData](#) ()

Default constructor.

3.15.2.2 [ServiceData::ServiceData](#) (const [ServiceData](#) & *src*)

Copying constructor.

Parameters

<i>src</i>	Object being copied.
------------	----------------------

3.15.2.3 ServiceData::ServiceData (unsigned int *id*, std::string *name*, unsigned *operationalDays*)

Constructor.

Parameters

<i>name</i>	Name of route.
<i>id</i>	id of route.

3.15.2.4 ServiceData::~ServiceData ()

Destructor.

3.15.3 Member Function Documentation

3.15.3.1 unsigned ServiceData::getDays ()

Returns

return days value.

3.15.3.2 unsigned int ServiceData::getId ()

Returns

Returns id value.

3.15.3.3 std::string ServiceData::getName ()

Returns

return name value.

3.15.3.4 ServiceData & ServiceData::operator= (const ServiceData & *src*)

Assignment operator.

Parameters

<i>src</i>	Reference object being assigned.
------------	----------------------------------

Returns

Reference to self.

3.15.3.5 bool ServiceData::operator== (const ServiceData *src*)

Comparison operator.

Parameters

<i>src</i>	Reference to compared object.
------------	-------------------------------

Returns

True if name and is are equal.

3.15.4 Friends And Related Function Documentation

3.15.4.1 `std::ostream& operator<< (std::ostream & output, const ServiceData & src)` [friend]

Helper output function.

Parameters

<i>output</i>	Stream
<i>src</i>	Outputed object.

Returns

Reference to given stream.

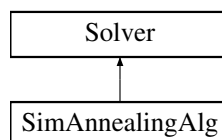
The documentation for this class was generated from the following files:

- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/ServiceData.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/ServiceData.cpp

3.16 SimAnnealingAlg Class Reference

```
#include <SimAnnealingAlg.h>
```

Inheritance diagram for SimAnnealingAlg:



Public Member Functions

- **SimAnnealingAlg** (long long seed)
- virtual `Route * solve` (const `Network *n`, `Node *start`, `Node *end`, `Time t`)
- void **setParams** (double Tstart, double Tend, unsigned int k, double alpha, unsigned int allowedChange↵ Number, unsigned int changePunishment)
- virtual const std::string & **getName** () const
- std::vector< unsigned > **getWeights** () const
- std::vector< unsigned > **getPunishments** () const
- std::vector< unsigned > **getBestPosWeights** () const
- void **setSeed** (long long seed)

3.16.1 Detailed Description

Simulated Annealing Algorithm used for finding routes. See doc folder for more information.

3.16.2 Member Function Documentation

3.16.2.1 `Route * SimAnnealingAlg::solve (const Network * n, Node * start, Node * end, Time t)` [virtual]

Method used in [Network](#) class for finding best connection between points.

Parameters

<code>n</code>	Pointer to Network in which Route is being searched for.
----------------	--

Returns

Pointer to found [Route](#), NULL if no route can be found.

Implements [Solver](#).

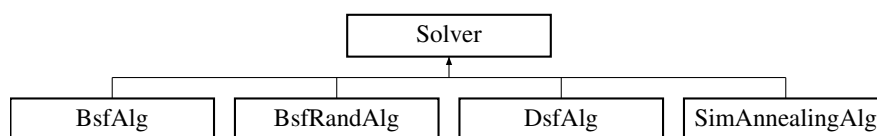
The documentation for this class was generated from the following files:

- /home/vka/Programming/C/workspace/RouteFinder/src/algorithm/SimAnnealingAlg.h
- /home/vka/Programming/C/workspace/RouteFinder/src/algorithm/SimAnnealingAlg.cpp

3.17 Solver Class Reference

```
#include <Solver.h>
```

Inheritance diagram for Solver:



Public Member Functions

- virtual [Route](#) * `solve` (const [Network](#) *n, [Node](#) *start, [Node](#) *end, [Time](#) t)=0
- virtual const std::string & `getName` () const =0

3.17.1 Detailed Description

wrapper class for solver algorithm. Those shall inherit from [Solver](#) class. [Solver](#) needs to implement solve method, which gets [Network](#) map as

```
USAGE: int main() { Solver * solver; solver = new SimAnnealingAlg; solver->setParams(...); Network net = new Network; ... net.setSolver(solver); //in Network: // this->solver->solve(); }
```

3.17.2 Member Function Documentation

3.17.2.1 `virtual Route* Solver::solve (const Network * n, Node * start, Node * end, Time t)` [pure virtual]

Method used in [Network](#) class for finding best connection between points. This method need to be implemented in any class inheriting from [Solver](#) class.

Parameters

<i>n</i>	Pointer to Network in which Route is being searched for.
----------	--

Returns

Pointer to found [Route](#), NULL if no route can be found.

Implemented in [SimAnnealingAlg](#), [BsfAlg](#), [BsfRandAlg](#), and [DsfAlg](#).

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

- /home/vka/Programming/C/workspace/RouteFinder/src/algorithm/Solver.h
- /home/vka/Programming/C/workspace/RouteFinder/src/algorithm/Solver.cpp

3.18 StopData Class Reference

```
#include <StopData.h>
```

Public Member Functions

- [StopData](#) ()
- [StopData](#) (const [StopData](#) &src)
- [StopData](#) (std::string name, unsigned int id, double lat, double lng)
- [~StopData](#) ()
- [StopData](#) & [operator=](#) (const [StopData](#) &src)
- bool [operator==](#) (const [StopData](#) src)
- std::string [getName](#) () const
- unsigned int [getId](#) () const
- double [getLat](#) () const
- double [getLng](#) () const

Friends

- std::ostream & [operator<<](#) (std::ostream &output, const [StopData](#) &src)

3.18.1 Detailed Description

Class containing data about routes when database is being created. Then converted into [Route](#) object.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 StopData::StopData ()

Default constructor.

3.18.2.2 StopData::StopData (const StopData & src)

Copying constructor.

Parameters

<i>src</i>	Object being copied.
------------	----------------------

3.18.2.3 StopData::StopData (std::string *name*, unsigned int *id*, double *lat*, double *lng*)

Constructor.

Parameters

<i>name</i>	Name of route.
<i>id</i>	id of route.
<i>lat,lng</i>	coords of stop.

3.18.2.4 StopData::~~StopData ()

Destructor.

3.18.3 Member Function Documentation

3.18.3.1 unsigned int StopData::getId () const

Returns

return id value.

3.18.3.2 double StopData::getLat () const

Returns

return latitude value.

3.18.3.3 double StopData::getLng () const

Returns

return longitude value.

3.18.3.4 std::string StopData::getName () const

Returns

Returns name value.

3.18.3.5 StopData & StopData::operator= (const StopData & *src*)

Assignment operator.

Parameters

<i>src</i>	Reference object being assigned.
------------	----------------------------------

Returns

Reference to self.

3.18.3.6 bool StopData::operator== (const StopData src)

Comparison operator.

Parameters

<i>src</i>	Reference to compared object.
------------	-------------------------------

Returns

True if name and is are equal.

3.18.4 Friends And Related Function Documentation**3.18.4.1 std::ostream& operator<< (std::ostream & output, const StopData & src) [friend]**

Helper output function.

Parameters

<i>output</i>	Stream
<i>src</i>	Outputed object.

Returns

Reference to given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/StopData.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/StopData.cpp

3.19 StopTimeData Class Reference

```
#include <StopTimeData.h>
```

Public Member Functions

- [StopTimeData](#) ()
- [StopTimeData](#) (const [StopTimeData](#) &src)
- [StopTimeData](#) (unsigned int id, unsigned int stopID, unsigned int serviceID, unsigned int tripID, std::string name)
- [~StopTimeData](#) ()
- [StopTimeData](#) & [operator=](#) (const [StopTimeData](#) &src)
- bool [operator==](#) (const [StopTimeData](#) src)
- unsigned int [getId](#) () const
- unsigned int [getServiceId](#) () const

- unsigned int [getStopId](#) () const
- unsigned int [getTripId](#) () const
- Time [getStopTime](#) () const
- std::string [getName](#) () const

Friends

- std::ostream & [operator<<](#) (std::ostream &output, const [StopTimeData](#) &src)

3.19.1 Detailed Description

Class containing data about stopTimes when database is being created.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 StopTimeData::StopTimeData ()

Default constructor.

3.19.2.2 StopTimeData::StopTimeData (const StopTimeData & src)

Copying constructor.

Parameters

<i>src</i>	Object being copied.
------------	----------------------

3.19.2.3 StopTimeData::StopTimeData (unsigned int *id*, unsigned int *stopID*, unsigned int *serviceID*, unsigned int *tripID*, std::string *name*)

Constructor.

Parameters

<i>name</i>	Name of route.
<i>id</i>	id of route.
<i>stopID</i>	ID of StopData object of which timetable info contains this object.
<i>serviceID</i>	id of ServiceData object.
<i>tripID</i>	TripData object being mentioned.

3.19.2.4 StopTimeData::~StopTimeData ()

Destructor.

3.19.3 Member Function Documentation

3.19.3.1 unsigned int StopTimeData::getId () const

Returns

Returns id value.

3.19.3.2 `std::string StopTimeData::getName () const`

Returns

Returns name value.

3.19.3.3 `unsigned int StopTimeData::getServiceId () const`

Returns

Returns [ServiceData](#) id value.

3.19.3.4 `unsigned int StopTimeData::getStopId () const`

Returns

Returns [StopData](#) id value.

3.19.3.5 `Time StopTimeData::getStopTime () const`

Returns

Returns [Time](#) value.

3.19.3.6 `unsigned int StopTimeData::getTripId () const`

Returns

Returns [TripData](#) id value.

3.19.3.7 `StopTimeData & StopTimeData::operator= (const StopTimeData & src)`

Assignment operator.

Parameters

<i>src</i>	Reference object being assigned.
------------	----------------------------------

Returns

Reference to self.

3.19.3.8 `bool StopTimeData::operator== (const StopTimeData src)`

Comparison operator.

Parameters

<i>src</i>	Reference to compared object.
------------	-------------------------------

Returns

True if name and is are equal.

3.19.4 Friends And Related Function Documentation

3.19.4.1 `std::ostream& operator<< (std::ostream & output, const StopTimeData & src)` [*friend*]

Helper output function.

Parameters

<i>output</i>	Stream
<i>src</i>	Outputed object.

Returns

Reference to given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/StopTimeData.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/StopTimeData.cpp

3.20 Tester Class Reference

Public Member Functions

- **Tester** ([Network](#) *n)
- void **makeTests** ([Node](#) *from, [Node](#) *to, std::string directory, long long seed=0)
- void **setDefaults** (double Tstart, double Tend, unsigned int k, double alpha, unsigned int allowedChange↵
Number, unsigned int changePunishment)

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

- /home/vka/Programming/C/workspace/RouteFinder/src/tester/Tester.h
- /home/vka/Programming/C/workspace/RouteFinder/src/tester/Tester.cpp

3.21 Time Class Reference

```
#include <Time.h>
```

Public Member Functions

- [Time](#) ()
- [Time](#) (unsigned int hour, unsigned int minute)
- [Time](#) (unsigned int minutes)
- [Time](#) (const [Time](#) &src)
- [~Time](#) ()
- [Time](#) & **operator=** (const [Time](#) &src)
- bool **operator==** (const [Time](#) &src) const
- bool **operator!=** (const [Time](#) &src) const
- bool **operator>** (const [Time](#) src)
- bool **operator>=** (const [Time](#) src)
- bool **operator<** (const [Time](#) src)
- bool **operator<=** (const [Time](#) src)
- [Time](#) **operator+** (const [Time](#) src)
- [Time](#) **operator-** (const [Time](#) src)
- [operator int](#) ()

Friends

- std::ostream & **operator<<** (std::ostream &output, const [Time](#) src)

3.21.1 Detailed Description

Class providing time functions used in timetable generation.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 Time::Time ()

Default constructor.

3.21.2.2 Time::Time (unsigned int *hour*, unsigned int *minute*)

Constructor.

Parameters

<i>hour</i>	Hour.
<i>minute</i>	Minute.

3.21.2.3 Time::Time (unsigned int *minutes*)

Constructor creating object pointing to time minutes ahead of midnight.

Parameters

<i>minutes</i>	Minutes from midnight.
----------------	------------------------

3.21.2.4 Time::Time (const Time & *src*)

Copying constructor.

Parameters

<i>src</i>	Object being copied.
------------	----------------------

3.21.2.5 Time::~Time ()

Destructor.

3.21.3 Member Function Documentation

3.21.3.1 Time::operator int ()

Converts [Time](#) to int.

3.21.3.2 bool Time::operator!= (const Time & *src*) const

Comparison operator.

Parameters

<i>src</i>	Reference to compared object.
------------	-------------------------------

Returns

True if given hours are not equal.

3.21.3.3 Time Time::operator+ (const Time *src*)

Addition operator. Creates object pointing to time created by adding two given hours.

Parameters

<i>src</i>	given Time object.
------------	------------------------------------

Returns

[Time](#) object.

3.21.3.4 Time Time::operator- (const Time *src*)

Subtraction operator. Creates object pointing to time created by subtracting two given hours.

Parameters

<i>src</i>	given Time object.
------------	------------------------------------

Returns

[Time](#) object.

3.21.3.5 bool Time::operator< (const Time *src*)

Greater than operator.

Returns

True if given "this" time is sooner from midnight than *src* time.

3.21.3.6 Time & Time::operator= (const Time & *src*)

Assignment operator.

Parameters

<i>src</i>	Reference object being assigned.
------------	----------------------------------

Returns

Reference to self.

3.21.3.7 bool Time::operator== (const Time & *src*) const

Comparison operator.

Parameters

<i>src</i>	Reference to compared object.
------------	-------------------------------

Returns

True if given hours are equal.

3.21.3.8 bool Time::operator> (const Time *src*)

Greater than operator.

Returns

True if given "this" time is later from midnight than *src* time.

3.21.4 Friends And Related Function Documentation

3.21.4.1 std::ostream& operator<< (std::ostream & *output*, const Time *src*) [friend]

Helper output function.

Parameters

<i>output</i>	Stream
<i>src</i>	Outputed object.

Returns

Reference to given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/Time.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/Time.cpp

3.22 TripData Class Reference

```
#include <TripData.h>
```

Public Member Functions

- [TripData](#) ()
- [TripData](#) (const [TripData](#) &*src*)
- [TripData](#) (unsigned int *id*, unsigned int *routeId*, std::string *name*, std::vector< int > *stopSec*)
- [~TripData](#) ()
- [TripData](#) & [operator=](#) (const [TripData](#) &*src*)
- bool [operator==](#) (const [TripData](#) *src*)
- unsigned int [getId](#) () const
- unsigned int [getRouteId](#) () const
- std::vector< int > [getStopSec](#) () const
- std::string [getName](#) () const

Friends

- `std::ostream & operator<< (std::ostream &output, const TripData &src)`

3.22.1 Detailed Description

Class containing data about Trip when database is being created. Then converted into Trip object.

3.22.2 Constructor & Destructor Documentation

3.22.2.1 `TripData::TripData ()`

Default constructor.

3.22.2.2 `TripData::TripData (const TripData & src)`

Copying constructor.

Parameters

<i>src</i>	Object being copied.
------------	----------------------

3.22.2.3 `TripData::TripData (unsigned int id, unsigned int routeId, std::string name, std::vector< int > stopSec)`

Constructor.

Parameters

<i>name</i>	Name of route.
<i>id</i>	id of route.
<i>routeId</i>	RouteData id value.
<i>stopSec</i>	Vector containing stop sequence.

3.22.2.4 `TripData::~~TripData ()`

Destructor.

3.22.3 Member Function Documentation

3.22.3.1 `unsigned int TripData::getId () const`

Returns

Returns id value.

3.22.3.2 `std::string TripData::getName () const`

Returns

Returns name value.

3.22.3.3 unsigned int TripData::getRouteId () const

Returns

Returns [RouteData](#) id value.

3.22.3.4 std::vector< int > TripData::getStopSec () const

Returns

Returns stop sequence in vector<int>.

3.22.3.5 TripData & TripData::operator= (const TripData & src)

Assignment operator.

Parameters

<i>src</i>	Reference object being assigned.
------------	----------------------------------

Returns

Reference to self.

3.22.3.6 bool TripData::operator== (const TripData src)

Comparison operator.

Parameters

<i>src</i>	Reference to compared object.
------------	-------------------------------

Returns

True if name and is are equal.

3.22.4 Friends And Related Function Documentation

3.22.4.1 std::ostream& operator<< (std::ostream & output, const TripData & src) [friend]

Helper output function.

Parameters

<i>output</i>	Stream
<i>src</i>	Outputed object.

Returns

Reference to given stream.

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

- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/TripData.h
- /home/vka/Programming/C/workspace/RouteFinder/src/db/lib/TripData.cpp

Index

- ~Network
 - Network, 18
- ~RouteData
 - RouteData, 29
- ~ServiceData
 - ServiceData, 31
- ~StopData
 - StopData, 35
- ~StopTimeData
 - StopTimeData, 37
- ~Time
 - Time, 41
- ~TripData
 - TripData, 44

- addEdge
 - Route, 24

- begin
 - Route, 25

- BsfAlg, 5
 - solve, 5
- BsfRandAlg, 6
 - solve, 6

- Connection, 6
 - Connection, 7
 - getArrivalTime, 7
 - getDepartureTime, 7
 - getTripID, 7
 - operator<<, 8
 - operator=, 7

- DataBase, 8
 - DataBase, 9
 - isValid, 9
 - LoadMethod, 9
 - routes, 9
 - services, 9
 - stopTimes, 9
 - stops, 9
 - trips, 9

- DataReader, 10
 - readRoutes, 10
 - readServices, 10
 - readStopTimes, 11
 - readStops, 10
 - readTrips, 11

- DsfAlg, 11
 - solve, 12

- Edge, 12
 - Edge, 12
 - getEndNode, 13
 - getID, 13
 - getStartNode, 13
 - operator!=, 13
 - operator<, 13
 - operator<<, 14
 - operator=, 13
 - operator==, 14
- edgePointerCompare, 14
- end
 - Route, 25

- findRouteBetween
 - Network, 18

- GTFSReader, 15
 - getRoutes, 15
 - getServices, 15
 - getStopTimes, 15
 - getStops, 15
 - getTrips, 15
 - operator<<, 17
 - readGTFS, 15

- getAllEdges
 - Network, 18

- getAllNodes
 - Network, 18

- getArrivalTime
 - Connection, 7

- getChangeNumber
 - Route, 25

- getConnectionsSequence
 - Route, 25

- getDays
 - ServiceData, 31

- getDepartureTime
 - Connection, 7

- getEdge
 - Network, 18

- getEdgesForNode
 - Network, 19

- getEndNode
 - Edge, 13
 - Route, 25

- getID
 - Edge, 13
 - Node, 22

- getId

- RouteData, 29
- ServiceData, 31
- StopData, 35
- StopTimeData, 37
- TripData, 44
- getLat
 - StopData, 35
- getLatitude
 - Node, 22
- getLength
 - Route, 25
- getLng
 - StopData, 35
- getLongitude
 - Node, 22
- getName
 - Node, 22
 - RouteData, 29
 - ServiceData, 31
 - StopData, 35
 - StopTimeData, 37
 - TripData, 44
- getNode
 - Network, 19
- getNodeCloseToPos
 - Network, 19
- getRouteId
 - TripData, 44
- getRoutes
 - GTFSReader, 15
- getServiceId
 - StopTimeData, 38
- getServices
 - GTFSReader, 15
- getStartNode
 - Edge, 13
 - Route, 25
- getStopId
 - StopTimeData, 38
- getStopSec
 - TripData, 45
- getStopTime
 - StopTimeData, 38
- getStopTimes
 - GTFSReader, 15
- getStops
 - GTFSReader, 15
- getTripID
 - Connection, 7
- getTripId
 - StopTimeData, 38
- getTrips
 - GTFSReader, 15
- getWeight
 - Route, 26
- incidenceMatrix
 - Network, 20
- isConnectionBetween
 - Route, 26
- isEdgeBetween
 - Network, 19
- isEdgeIn
 - Route, 26
- isNodeIn
 - Route, 26
- isValid
 - DataBase, 9
- LoadMethod
 - DataBase, 9
- Network, 17
 - ~Network, 18
 - findRouteBetween, 18
 - getAllEdges, 18
 - getAllNodes, 18
 - getEdge, 18
 - getEdgesForNode, 19
 - getNode, 19
 - getNodeCloseToPos, 19
 - incidenceMatrix, 20
 - isEdgeBetween, 19
 - Network, 18
 - operator<<, 20
 - setSolver, 19
- Node, 20
 - getID, 22
 - getLatitude, 22
 - getLongitude, 22
 - getName, 22
 - Node, 21
 - operator!=, 22
 - operator<, 22
 - operator<<, 23
 - operator=, 23
 - operator==, 23
- nodePointerCompare, 23
- operator int
 - Time, 41
- operator!=
 - Edge, 13
 - Node, 22
 - Time, 41
- operator<
 - Edge, 13
 - Node, 22
 - Time, 42
- operator<<
 - Connection, 8
 - Edge, 14
 - GTFSReader, 17
 - Network, 20
 - Node, 23
 - Route, 27
 - RouteData, 29
 - ServiceData, 32

- StopData, 36
- StopTimeData, 39
- Time, 43
- TripData, 45
- operator>
 - Time, 43
- operator+
 - Time, 42
- operator-
 - Time, 42
- operator=
 - Connection, 7
 - Edge, 13
 - Node, 23
 - RouteData, 29
 - ServiceData, 31
 - StopData, 35
 - StopTimeData, 38
 - Time, 42
 - TripData, 45
- operator==
 - Edge, 14
 - Node, 23
 - RouteData, 29
 - ServiceData, 31
 - StopData, 36
 - StopTimeData, 38
 - Time, 42
 - TripData, 45
- printRoute
 - Route, 26
- readGTFS
 - GTFSReader, 15
- readRoutes
 - DataReader, 10
- readServices
 - DataReader, 10
- readStopTimes
 - DataReader, 11
- readStops
 - DataReader, 10
- readTrips
 - DataReader, 11
- Route, 24
 - addEdge, 24
 - begin, 25
 - end, 25
 - getChangeNumber, 25
 - getConnectionsSequence, 25
 - getEndNode, 25
 - getLength, 25
 - getStartNode, 25
 - getWeight, 26
 - isConnectionBetween, 26
 - isEdgeIn, 26
 - isNodeIn, 26
 - operator<<, 27
 - printRoute, 26
 - Route, 24
 - switchEdge, 27
 - switchRoute, 27
 - validate, 27
- RouteData, 28
 - ~RouteData, 29
 - getId, 29
 - getName, 29
 - operator<<, 29
 - operator=, 29
 - operator==, 29
 - RouteData, 28
- routes
 - DataBase, 9
- ServiceData, 30
 - ~ServiceData, 31
 - getDays, 31
 - getId, 31
 - getName, 31
 - operator<<, 32
 - operator=, 31
 - operator==, 31
 - ServiceData, 30, 31
- services
 - DataBase, 9
- setSolver
 - Network, 19
- SimAnnealingAlg, 32
 - solve, 33
- solve
 - BsfAlg, 5
 - BsfRandAlg, 6
 - DsfAlg, 12
 - SimAnnealingAlg, 33
 - Solver, 33
- Solver, 33
 - solve, 33
- StopData, 34
 - ~StopData, 35
 - getId, 35
 - getLat, 35
 - getLng, 35
 - getName, 35
 - operator<<, 36
 - operator=, 35
 - operator==, 36
 - StopData, 34, 35
- StopTimeData, 36
 - ~StopTimeData, 37
 - getId, 37
 - getName, 37
 - getServiceId, 38
 - getStopId, 38
 - getStopTime, 38
 - getTripId, 38
 - operator<<, 39
 - operator=, 38

- operator==, [38](#)
 - StopTimeData, [37](#)
- stopTimes
 - DataBase, [9](#)
- stops
 - DataBase, [9](#)
- switchEdge
 - Route, [27](#)
- switchRoute
 - Route, [27](#)
- Tester, [40](#)
- Time, [40](#)
 - ~Time, [41](#)
 - operator int, [41](#)
 - operator!=, [41](#)
 - operator<, [42](#)
 - operator<<, [43](#)
 - operator>, [43](#)
 - operator+, [42](#)
 - operator-, [42](#)
 - operator=, [42](#)
 - operator==, [42](#)
 - Time, [41](#)
- TripData, [43](#)
 - ~TripData, [44](#)
 - getId, [44](#)
 - getName, [44](#)
 - getRouteId, [44](#)
 - getStopSec, [45](#)
 - operator<<, [45](#)
 - operator=, [45](#)
 - operator==, [45](#)
 - TripData, [44](#)
- trips
 - DataBase, [9](#)
- validate
 - Route, [27](#)