Deliverable 3

1.0

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 GPS Namespace Reference

Classes

- class Route
- class Track

Chapter 6

Class Documentation

6.1 GPS::Route Class Reference

```
#include <route.h>
```

Inheritance diagram for GPS::Route:

Public Member Functions

- Route (std::string source, bool isFileName, metres granularity=20)
- virtual void setGranularity (metres)
- std::string name () const
- unsigned int numPositions () const
- metres totalLength () const
- metres netLength () const
- metres totalHeightGain () const
- metres netHeightGain () const
- degrees maxGradient () const
- degrees minGradient () const
- degrees steepestGradient () const
- degrees minLatitude () const
- degrees maxLatitude () const
- degrees minLongitude () const
- degrees maxLongitude () const
- metres minElevation () const
- metres maxElevation () const
- Position operator[ ] (unsigned int) const
- Position findPosition (std::string soughtName) const
- std::string findNameOf (Position) const
- unsigned int timesVisited (std::string soughtName) const
- unsigned int timesVisited (Position) const
- bool containsCycles () const

Protected Member Functions

- Route ()
- bool areSameLocation (Position, Position) const

Protected Attributes

- std::vector< Position > positions
- std::vector< std::string > positionNames
- std::string routeName
- metres routeLength
- metres granularity

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Route() [1/2]

```
Route::Route (
             std::string source,
             bool isFileName,
             metres granularity = 20 )
```

6.1.1.2 Route() [2/2]

```
GPS::Route::Route ( )  [inline], [protected]
```

6.1.2 Member Function Documentation

6.1.2.1 areSameLocation()

```
bool Route::areSameLocation (
             Position p1,
             Position p2 ) const  [protected]
```

6.1.2.2 containsCycles()

```
bool Route::containsCycles ( ) const
```

6.1.2.3 findNameOf()

```
std::string Route::findNameOf (
             Position soughtPos ) const
```

6.1.2.4 findPosition()

```
Position Route::findPosition (
             std::string soughtName ) const
```

6.1.2.5 maxElevation()

```
metres Route::maxElevation ( ) const
```

6.1.2.6 maxGradient()

```
degrees Route::maxGradient ( ) const
```

6.1.2.7 maxLatitude()

```
degrees Route::maxLatitude ( ) const
```

6.1.2.8 maxLongitude()

```
degrees Route::maxLongitude ( ) const
```

6.1.2.9 minElevation()

```
metres Route::minElevation ( ) const
```

6.1.2.10 minGradient()

```
degrees Route::minGradient ( ) const
```

6.1.2.11 minLatitude()

```
degrees Route::minLatitude ( ) const
```

6.1.2.12 minLongitude()

```
degrees Route::minLongitude ( ) const
```

6.1.2.13 name()

```
std::string Route::name ( ) const
```

6.1.2.14 netHeightGain()

```
metres Route::netHeightGain ( ) const
```

6.1.2.15 netLength()

```
metres Route::netLength ( ) const
```

6.1.2.16 numPositions()

```
unsigned int Route::numPositions ( ) const
```

6.1.2.17 operator[]()

```
Position Route::operator[] (
             unsigned int idx ) const
```

6.1.2.18 setGranularity()

```
void Route::setGranularity (
             metres  )  [virtual]
```

Reimplemented in GPS::Track.

6.1.2.19 steepestGradient()

```
degrees Route::steepestGradient ( ) const
```

6.1.2.20 timesVisited() [1/2]

```
unsigned int Route::timesVisited (
             Position soughtPos ) const
```

6.1.2.21 timesVisited() [2/2]

```
unsigned int Route::timesVisited (
             std::string soughtName ) const
```

6.1.2.22 totalHeightGain()

```
metres Route::totalHeightGain ( ) const
```

6.1.2.23 totalLength()

```
metres Route::totalLength ( ) const
```

6.1.3 Member Data Documentation

6.1.3.1 granularity

```
metres GPS::Route::granularity  [protected]
```

6.1.3.2 positionNames

```
std::vector<std::string> GPS::Route::positionNames  [protected]
```

6.1.3.3 positions

```
std::vector<Position> GPS::Route::positions  [protected]
```

6.1.3.4 routeLength

```
metres GPS::Route::routeLength  [protected]
```

6.1.3.5 routeName

```
std::string GPS::Route::routeName  [protected]
```

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

- route.h
- route.cpp

6.2 GPS::Track Class Reference

```
#include <track.h>
```

Inheritance diagram for GPS::Track:

Collaboration diagram for GPS::Track:

Public Member Functions

- Track (std::string source, bool isFileName, metres granularity=10)
- void setGranularity (metres) override
- seconds totalTime () const
- seconds travellingTime () const
- seconds restingTime () const
- seconds longestRest () const
- speed maxSpeed () const
- speed averageSpeed (bool includeRests) const
- speed maxRateOfAscent () const
- speed maxRateOfDescent () const

Protected Attributes

- std::vector< seconds > arrived
- std::vector< seconds > departed

Additional Inherited Members

6.2.1 Constructor & Destructor Documentation

6.2.1.1 Track()

```
Track::Track (
            std::string source,
            bool isFileName,
            metres granularity = 10 )
```

6.2.2 Member Function Documentation

6.2.2.1 averageSpeed()

```
speed Track::averageSpeed (
            bool includeRests ) const
```

6.2.2.2 longestRest()

```
seconds Track::longestRest ( ) const
```

6.2.2.3 maxRateOfAscent()

```
speed Track::maxRateOfAscent ( ) const
```

6.2.2.4 maxRateOfDescent()

```
speed Track::maxRateOfDescent ( ) const
```

6.2.2.5 maxSpeed()

```
speed Track::maxSpeed ( ) const
```

6.2.2.6 restingTime()

```
seconds Track::restingTime ( ) const
```

6.2.2.7 setGranularity()

```
void Track::setGranularity (
            metres  )  [override], [virtual]
```

Reimplemented from GPS::Route.

6.2.2.8 totalTime()

```
seconds Track::totalTime ( ) const
```

6.2.2.9 travellingTime()

```
seconds Track::travellingTime ( ) const
```

6.2.3 Member Data Documentation

6.2.3.1 arrived

```
std::vector<seconds> GPS::Track::arrived  [protected]
```

6.2.3.2 departed

```
std::vector<seconds> GPS::Track::departed  [protected]
```

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

- track.h
- track.cpp

Chapter 7

File Documentation

7.1 gpx-tests.cpp File Reference

```
#include <boost/test/unit_test.hpp>
```

Include dependency graph for gpx-tests.cpp:

7.2 main.cpp File Reference

```
#include <iostream>
#include "logs.h"
#include "route.h"
#include "track.h"
```

Include dependency graph for main.cpp:

Functions

- void testRoute (std::string fileName)
- void testTrack (std::string fileName)
- int main ()

7.2.1 Function Documentation

7.2.1.1 main()

```
int main ( )
```

7.2.1.2 testRoute()

```
void testRoute (
             std::string fileName )
```

7.2.1.3 testTrack()

```
void testTrack (
             std::string fileName )
```

7.3 route.cpp File Reference

```
#include <sstream>
#include <fstream>
#include <iostream>
#include <cassert>
#include <cmath>
#include <algorithm>
#include <iterator>
#include <stdexcept>
#include "geometry.h"
#include "xml/element.h"
#include "xml/parser.h"
#include "route.h"
```

Include dependency graph for route.cpp:

7.4 route.h File Reference

```
#include <string>
#include <vector>
#include "types.h"
#include "position.h"
#include "xml/parser.h"
```

Include dependency graph for route.h: This graph shows which files directly or indirectly include this file:

Classes

- class GPS::Route

Namespaces

- GPS

7.5 timesVisited(string).cpp File Reference

```
#include <boost/test/unit_test.hpp>
#include "logs.h"
#include "route.h"
```

Include dependency graph for timesVisited(string).cpp:

Functions

- BOOST_AUTO_TEST_CASE (singleton_route)

  A simple route with one point and one name to check.
- BOOST_AUTO_TEST_CASE (position_not_visited)

  A simple route with one point and checking the number of times for a position that wasn't visited.
- BOOST_AUTO_TEST_CASE (singleton_route_with_spaces)

  A simple route with one point and one name to check, with leading and trailing spaces to check the method matches the constructor.
- BOOST_AUTO_TEST_CASE (bad_input_string)

  A simple route with one point and checking invalid_argument is thrown when a blank string is passed in.
- BOOST_AUTO_TEST_CASE (two_consecutive_visits)

  A simple route with two positions that are the same location.
- BOOST_AUTO_TEST_CASE (one_name_multiple_positions)

  A complex route where many positions share the same name.
- BOOST_AUTO_TEST_CASE (one_position_many_visits)

  A complex route where one position is visited many times.
- BOOST_AUTO_TEST_CASE (one_name_many_positions_and_many_visits)

  A complex route where there are many positions with the same name, visited multiple times.
- BOOST_AUTO_TEST_CASE (one_position_many_names)

  A complex route where there is one position visited multiple times, but with different names.

Variables

- const bool isFileName = false

  all data for this test suite is passed in as strings, not files.

7.5.1 Function Documentation

7.5.1.1 BOOST_AUTO_TEST_CASE() [1/9]

```
BOOST_AUTO_TEST_CASE (
          bad_input_string )
```

A simple route with one point and checking invalid_argument is thrown when a blank string is passed in.

7.5.1.2 BOOST_AUTO_TEST_CASE() [2/9]

```
BOOST_AUTO_TEST_CASE (
            one_name_many_positions_and_many_visits  )
```

A complex route where there are many positions with the same name, visited multiple times.

7.5.1.3 BOOST_AUTO_TEST_CASE() [3/9]

```
BOOST_AUTO_TEST_CASE (
            one_name_multiple_positions  )
```

A complex route where many positions share the same name.

7.5.1.4 BOOST_AUTO_TEST_CASE() [4/9]

```
BOOST_AUTO_TEST_CASE (
            one_position_many_names  )
```

A complex route where there is one position visited multiple times, but with different names.

7.5.1.5 BOOST_AUTO_TEST_CASE() [5/9]

```
BOOST_AUTO_TEST_CASE (
            one_position_many_visits  )
```

A complex route where one position is visited many times.

7.5.1.6 BOOST_AUTO_TEST_CASE() [6/9]

```
BOOST_AUTO_TEST_CASE (
            position_not_visited  )
```

A simple route with one point and checking the number of times for a position that wasn't visited.

7.5.1.7 BOOST_AUTO_TEST_CASE() [7/9]

```
BOOST_AUTO_TEST_CASE (
            singleton_route  )
```

A simple route with one point and one name to check.

7.5.1.8 BOOST_AUTO_TEST_CASE() [8/9]

```
BOOST_AUTO_TEST_CASE (
            singleton_route_with_spaces  )
```

A simple route with one point and one name to check, with leading and trailing spaces to check the method matches the constructor.

7.5.1.9 BOOST_AUTO_TEST_CASE() [9/9]

```
BOOST_AUTO_TEST_CASE (
            two_consecutive_visits  )
```

A simple route with two positions that are the same location.

7.5.2 Variable Documentation

7.5.2.1 isFileName

```
const bool isFileName = false
```

all data for this test suite is passed in as strings, not files.

7.6 track.cpp File Reference

```
#include <sstream>
#include <fstream>
#include <iostream>
#include <cassert>
#include <cmath>
#include <stdexcept>
#include "geometry.h"
#include "xml/element.h"
#include "xml/parser.h"
#include "track.h"
```

Include dependency graph for track.cpp:

7.7 track.h File Reference

```
#include <string>
#include <vector>
#include "types.h"
#include "position.h"
#include "route.h"
#include "xml/parser.h"
```

Include dependency graph for track.h: This graph shows which files directly or indirectly include this file:

Classes

- class GPS::Track

Namespaces

- GPS

Index