

## Deliverable 3

1.0

Generated by Doxygen 1.8.17



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List . . . . .	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy . . . . .	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List . . . . .	5
<b>4 File Index</b>	<b>7</b>
4.1 File List . . . . .	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 GPS Namespace Reference . . . . .	9
<b>6 Class Documentation</b>	<b>11</b>
6.1 GPS::Route Class Reference . . . . .	11
6.1.1 Constructor & Destructor Documentation . . . . .	12
6.1.1.1 Route() [1/2] . . . . .	12
6.1.1.2 Route() [2/2] . . . . .	12
6.1.2 Member Function Documentation . . . . .	12
6.1.2.1 areSameLocation() . . . . .	12
6.1.2.2 containsCycles() . . . . .	12
6.1.2.3 findNameOf() . . . . .	13
6.1.2.4 findPosition() . . . . .	13
6.1.2.5 maxElevation() . . . . .	13
6.1.2.6 maxGradient() . . . . .	13
6.1.2.7 maxLatitude() . . . . .	13
6.1.2.8 maxLongitude() . . . . .	13
6.1.2.9 minElevation() . . . . .	13
6.1.2.10 minGradient() . . . . .	14
6.1.2.11 minLatitude() . . . . .	14
6.1.2.12 minLongitude() . . . . .	14
6.1.2.13 name() . . . . .	14
6.1.2.14 netHeightGain() . . . . .	14
6.1.2.15 netLength() . . . . .	14
6.1.2.16 numPositions() . . . . .	14
6.1.2.17 operator[]() . . . . .	14
6.1.2.18 setGranularity() . . . . .	15
6.1.2.19 steepestGradient() . . . . .	15
6.1.2.20 timesVisited() [1/2] . . . . .	15
6.1.2.21 timesVisited() [2/2] . . . . .	15
6.1.2.22 totalHeightGain() . . . . .	15
6.1.2.23 totalLength() . . . . .	15

6.1.3 Member Data Documentation	15
6.1.3.1 granularity	16
6.1.3.2 positionNames	16
6.1.3.3 positions	16
6.1.3.4 routeLength	16
6.1.3.5 routeName	16
6.2 GPS::Track Class Reference	16
6.2.1 Constructor & Destructor Documentation	17
6.2.1.1 Track()	17
6.2.2 Member Function Documentation	17
6.2.2.1 averageSpeed()	17
6.2.2.2 longestRest()	17
6.2.2.3 maxRateOfAscent()	18
6.2.2.4 maxRateOfDescent()	18
6.2.2.5 maxSpeed()	18
6.2.2.6 restingTime()	18
6.2.2.7 setGranularity()	18
6.2.2.8 totalTime()	18
6.2.2.9 travellingTime()	18
6.2.3 Member Data Documentation	18
6.2.3.1 arrived	19
6.2.3.2 departed	19
<b>7 File Documentation</b>	<b>21</b>
7.1 gpx-tests.cpp File Reference	21
7.2 main.cpp File Reference	21
7.2.1 Function Documentation	21
7.2.1.1 main()	21
7.2.1.2 testRoute()	22
7.2.1.3 testTrack()	22
7.3 route.cpp File Reference	22
7.4 route.h File Reference	22
7.5 timesVisited(string).cpp File Reference	23
7.5.1 Function Documentation	23
7.5.1.1 BOOST_AUTO_TEST_CASE() [1/9]	23
7.5.1.2 BOOST_AUTO_TEST_CASE() [2/9]	24
7.5.1.3 BOOST_AUTO_TEST_CASE() [3/9]	24
7.5.1.4 BOOST_AUTO_TEST_CASE() [4/9]	24
7.5.1.5 BOOST_AUTO_TEST_CASE() [5/9]	24
7.5.1.6 BOOST_AUTO_TEST_CASE() [6/9]	24
7.5.1.7 BOOST_AUTO_TEST_CASE() [7/9]	24
7.5.1.8 BOOST_AUTO_TEST_CASE() [8/9]	25

---

7.5.1.9 BOOST_AUTO_TEST_CASE() [9/9] . . . . .	25
7.5.2 Variable Documentation . . . . .	25
7.5.2.1 isFileName . . . . .	25
7.6 track.cpp File Reference . . . . .	25
7.7 track.h File Reference . . . . .	25
<b>Index</b>	<b>27</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">GPS</a> . . . . .	9
-------------------------------	---





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

GPS::Route . . . . .	11
GPS::Track . . . . .	16



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">GPS::Route</a>	.....	<a href="#">11</a>
<a href="#">GPS::Track</a>	.....	<a href="#">16</a>



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">gpx-tests.cpp</a>	21
<a href="#">main.cpp</a>	21
<a href="#">route.cpp</a>	22
<a href="#">route.h</a>	22
<a href="#">timesVisited(string).cpp</a>	23
<a href="#">track.cpp</a>	25
<a href="#">track.h</a>	25



## 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

- areSameLocation
  - GPS::Route, 12
- arrived
  - GPS::Track, 18
- averageSpeed
  - GPS::Track, 17
- BOOST\_AUTO\_TEST\_CASE
  - timesVisited(string).cpp, 23–25
- containsCycles
  - GPS::Route, 12
- departed
  - GPS::Track, 19
- findNameOf
  - GPS::Route, 12
- findPosition
  - GPS::Route, 13
- GPS, 9
- GPS::Route, 11
  - areSameLocation, 12
  - containsCycles, 12
  - findNameOf, 12
  - findPosition, 13
  - granularity, 15
  - maxElevation, 13
  - maxGradient, 13
  - maxLatitude, 13
  - maxLongitude, 13
  - minElevation, 13
  - minGradient, 13
  - minLatitude, 14
  - minLongitude, 14
  - name, 14
  - netHeightGain, 14
  - netLength, 14
  - numPositions, 14
  - operator[], 14
  - positionNames, 16
  - positions, 16
  - Route, 12
  - routeLength, 16
  - routeName, 16
  - setGranularity, 14
  - steepestGradient, 15
  - timesVisited, 15
  - totalHeightGain, 15
  - totalLength, 15
- GPS::Track, 16
  - arrived, 18
  - averageSpeed, 17
  - departed, 19
  - longestRest, 17
  - maxRateOfAscent, 17
  - maxRateOfDescent, 18
  - maxSpeed, 18
  - restingTime, 18
  - setGranularity, 18
  - totalTime, 18
  - Track, 17
  - travellingTime, 18
- gpx-tests.cpp, 21
- granularity
  - GPS::Route, 15
- isFileName
  - timesVisited(string).cpp, 25
- longestRest
  - GPS::Track, 17
- main
  - main.cpp, 21
- main.cpp, 21
  - main, 21
  - testRoute, 21
  - testTrack, 22
- maxElevation
  - GPS::Route, 13
- maxGradient
  - GPS::Route, 13
- maxLatitude
  - GPS::Route, 13
- maxLongitude
  - GPS::Route, 13
- maxRateOfAscent
  - GPS::Track, 17
- maxRateOfDescent
  - GPS::Track, 18
- maxSpeed
  - GPS::Track, 18
- minElevation
  - GPS::Route, 13
- minGradient
  - GPS::Route, 13
- minLatitude
  - GPS::Route, 14
- minLongitude

- GPS::Route, [14](#)
- name
  - GPS::Route, [14](#)
- netHeightGain
  - GPS::Route, [14](#)
- netLength
  - GPS::Route, [14](#)
- numPositions
  - GPS::Route, [14](#)
- operator[]
  - GPS::Route, [14](#)
- positionNames
  - GPS::Route, [16](#)
- positions
  - GPS::Route, [16](#)
- restingTime
  - GPS::Track, [18](#)
- Route
  - GPS::Route, [12](#)
- route.cpp, [22](#)
- route.h, [22](#)
- routeLength
  - GPS::Route, [16](#)
- routeName
  - GPS::Route, [16](#)
- setGranularity
  - GPS::Route, [14](#)
  - GPS::Track, [18](#)
- steepestGradient
  - GPS::Route, [15](#)
- testRoute
  - main.cpp, [21](#)
- testTrack
  - main.cpp, [22](#)
- timesVisited
  - GPS::Route, [15](#)
- timesVisited(string).cpp, [23](#)
  - BOOST\_AUTO\_TEST\_CASE, [23–25](#)
  - isFileName, [25](#)
- totalHeightGain
  - GPS::Route, [15](#)
- totalLength
  - GPS::Route, [15](#)
- totalTime
  - GPS::Track, [18](#)
- Track
  - GPS::Track, [17](#)
- track.cpp, [25](#)
- track.h, [25](#)
- travellingTime
  - GPS::Track, [18](#)