# labo_11_schaufelberger_yannick_gallay_david

Generated by Doxygen 1.8.13

# Contents

# Chapter 1

# File Index

## 1.1 File List

Here is a list of all files with brief descriptions:

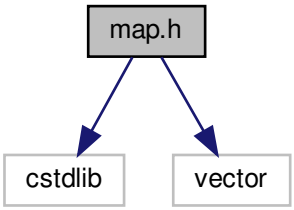
# Chapter 2

# File Documentation

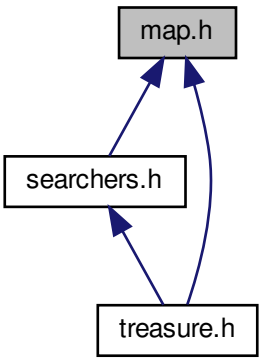
## 2.1 map.h File Reference

```
#include <cstdlib>
#include <vector>
```

Include dependency graph for map.h:




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



### Typedefs

- typedef std::vector< MapState > Axe
- typedef std::vector< Axe > Map

### Enumerations

- enum MapState {
  MS_OUT, MS_EARTH, MS_WATER, MS_TREASURE,
  MS_START }

### Functions

- size_t getHeight (const Map &map)
- size_t getWidth (const Map &map)
- Map getEmptyMap (size_t height, size_t width)
- MapState getMapValue (const Map &map, size_t x, size_t y)
- bool setMapValue (Map &map, size_t x, size_t y, MapState value)

  *sets the MapState at the given coordinates to the given value*
- bool addTreasure (Map &map, size_t height, size_t width)

  *if the MapState at the given coordinates is MS_EARTA, sets it to MS_TREASURE*
- void addRandomTreasure (Map &map)

  *sets a random valid cell to MS_TREASURE*
- bool addLake (Map &map, size_t originX, size_t originY, size_t radius)

  *if the given coordinates and radius are valid, adds a lake to the map*
- void addRandomLake (Map &map)

  *calls addRandomLake with the same map and a radius set to the third of the smallest size*
- void addRandomLake (Map &map, size_t maxRadius)

  *adds a lake at random coordinates with a random radius*
- bool addStart (Map &map, size_t x, size_t y)

  *if the MapState at the given coordinates is MS_EARTA, sets it to MS_START*
- void addRandomStart (Map &map)

  *calls addRandomStart with an x and y parameter*
- void addRandomStart (Map &map, size_t &x, size_t &y)

  *sets a random valid cell to MS_START.*
- Map initWorld (size_t height, size_t width, size_t &startX, size_t &startY)
- void displayWorld (const Map &map)

  *displays the map*

### Variables

- const int NUMBER_OF_LAKE = 3

#### 2.1.1 Typedef Documentation

#### 2.1.1.1 Axe

```
typedef std::vector<MapState> Axe
```

#### 2.1.1.2 Map

```
typedef std::vector<Axe> Map
```

### 2.1.2 Enumeration Type Documentation

#### 2.1.2.1 MapState

```
enum MapState
```

**Enumerator**

| | |
|---|---|
| MS_OUT | |
| MS_EARTH | |
| MS_WATER | |
| MS_TREASURE | |
| MS_START | |

### 2.1.3 Function Documentation

#### 2.1.3.1 addLake()

```
bool addLake (
            Map & map,
            size_t originX,
            size_t originY,
            size_t radius )
```

if the given coordinates and radius are valid, adds a lake to the map

**Parameters**

| | |
|---|---|
| *map* | |
| *originX* | |
| *originY* | |
| *radius* | |

**Returns**

true if success, false if not

#### 2.1.3.2 addRandomLake() [1/2]

```
void addRandomLake (
            Map & map )
```

calls addRandomLake with the same map and a radius set to the third of the smallest size

**Parameters**

| | |
|---|---|
| *map* | |

#### 2.1.3.3 addRandomLake() [2/2]

```
void addRandomLake (
            Map & map,
            size_t maxRadius )
```

adds a lake at random coordinates with a random radius

**Parameters**

| | |
|---|---|
| *map* | |
| *maxRadius* | |

#### 2.1.3.4 addRandomStart() [1/2]

```
void addRandomStart (
            Map & map )
```

calls addRandomStart with an x and y parameter

**Parameters**

| | |
|---|---|
| *map* | |

#### 2.1.3.5 addRandomStart() [2/2]

```
void addRandomStart (
            Map & map,
            size_t & x,
            size_t & y )
```

sets a random valid cell to MS_START.

**Parameters**

| | |
|---|---|
| *map* | |
| *x* | |
| *y* | |

#### 2.1.3.6 addRandomTreasure()

```
void addRandomTreasure (
            Map & map )
```

sets a random valid cell to MS_TREASURE

**Parameters**

| | |
|---|---|
| *map* | |

#### 2.1.3.7 addStart()

```
bool addStart (
            Map & map,
            size_t x,
            size_t y )
```

if the MapState at the given coordinates is MS_EARTA, sets it to MS_START

**Parameters**

| | |
|---|---|
| *map* | |
| *height* | |
| *width* | |

**Returns**

true if success, false if not

#### 2.1.3.8 addTreasure()

```
bool addTreasure (
            Map & map,
            size_t height,
            size_t width )
```

if the MapState at the given coordinates is MS_EARTA, sets it to MS_TREASURE

**Parameters**

| | |
|---|---|
| *map* | |
| *height* | |
| *width* | |

**Returns**

true if success, false if not

#### 2.1.3.9 displayWorld()

```
void displayWorld (
            const Map & map )
```

displays the map

**Parameters**

| | |
|---|---|
| *map* | |

#### 2.1.3.10 getEmptyMap()

```
Map getEmptyMap (
            size_t height,
            size_t width )
```

**Parameters**

| | |
|---|---|
| *height* | |
| *width* | |

**Returns**

an Map with the given sizes and filled with MS_EARTH

#### 2.1.3.11 getHeight()

```
size_t getHeight (
            const Map & map )
```

**Parameters**

| | |
|---|---|
| *map* | |

**Returns**

the height of the map aka the size of the vector<Axe>

#### 2.1.3.12 getMapValue()

```
MapState getMapValue (
            const Map & map,
            size_t x,
            size_t y )
```

**Parameters**

| | |
|---|---|
| *map* | |
| *x* | |
| *y* | |

**Returns**

the MapState at the given coordinates

#### 2.1.3.13 getWidth()

```
size_t getWidth (
            const Map & map )
```

**Parameters**

| | |
|---|---|
| *map* | |

**Returns**

the width of the map aka the size of the vector<MapState>

#### 2.1.3.14 initWorld()

```
Map initWorld (
            size_t height,
            size_t width,
            size_t & startX,
            size_t & startY )
```

**Parameters**

| | |
|---|---|
| *height* | the height of the map |
| *width* | the width of the map |
| *x* | the x coordinate of the MS_START |
| *y* | the y coordinate of the MS_START |

**Returns**

a map with NUMBER_OF_LAKE lakes, one start and one treasure

#### 2.1.3.15 setMapValue()

```
bool setMapValue (
            Map & map,
            size_t x,
            size_t y,
            MapState value )
```

sets the MapState at the given coordinates to the given value

**Parameters**

| | |
|---|---|
| *map* | |
| *x* | |
| *y* | |
| *value* | |

**Returns**

true if success, false if not

### 2.1.4 Variable Documentation

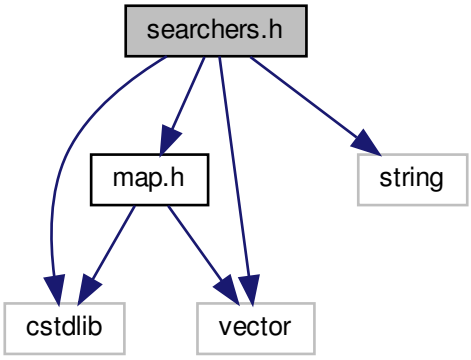
#### 2.1.4.1 NUMBER_OF_LAKE

```
const int NUMBER_OF_LAKE = 3
```

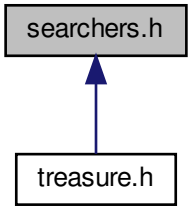
## 2.2 searchers.h File Reference

```
#include "map.h"
#include <cstdlib>
#include <vector>
#include <string>
```

Include dependency graph for searchers.h:




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




### Typedefs

- typedef std::vector< int > Searcher
- typedef std::vector< Searcher > SearcherList

### Enumerations

- enum ResearcherStatus {
  UNDEFINED, RICH, LOST, DROWNED,
  EXHAUSTED }

### Functions

- Searcher initSearcher (int stepsValue=0, ResearcherStatus statusValue=UNDEFINED)
- int getSteps (const Searcher &searcher)
- int getStatus (const Searcher &searcher)
- std::string getStatusString (const Searcher &searcher)
- bool setSteps (Searcher &searcher, int value)

  *sets the steps of a searcher to the given value*
- bool setStatus (Searcher &searcher, ResearcherStatus value)

  *sets the status of a searcher to the given value*
- bool displaySearcherList (SearcherList &list)

  *displays the information of every searcher in the list*

## 2.2.1 Typedef Documentation

#### 2.2.1.1 Searcher

```
typedef std::vector<int> Searcher
```

#### 2.2.1.2 SearcherList

```
typedef std::vector<Searcher> SearcherList
```

## 2.2.2 Enumeration Type Documentation

#### 2.2.2.1 ResearcherStatus

```
enum ResearcherStatus
```

**Enumerator**

| | |
|---|---|
| UNDEFINED | |
| RICH | |
| LOST | |
| DROWNED | |
| EXHAUSTED | |

### 2.2.3 Function Documentation

#### 2.2.3.1 displaySearcherList()

```
bool displaySearcherList (
            SearcherList & list )
```

displays the information of every searcher in the list

**Parameters**

| | |
|---|---|
| *list* | |

**Returns**

true if success, false if not

#### 2.2.3.2 getStatus()

```
int getStatus (
            const Searcher & searcher )
```

**Parameters**

| | |
|---|---|
| *searcher* | |

**Returns**

the status of a searcher

#### 2.2.3.3 getStatusString()

```
std::string getStatusString (
            const Searcher & searcher )
```

**Parameters**

| | |
|---|---|
| *searcher* | |

**Returns**

a string containing the status of a searcher

#### 2.2.3.4 getSteps()

```
int getSteps (
            const Searcher & searcher )
```

**Parameters**

| | |
|---|---|
| *searcher* | |

**Returns**

the amount of steps taken by a searcher

#### 2.2.3.5 initSearcher()

```
Searcher initSearcher (
            int stepsValue = 0,
            ResearcherStatus statusValue = UNDEFINED )
```

**Parameters**

| | |
|---|---|
| *stepsValue* | |
| *statusValue* | |

**Returns**

a searcher initialized with the given values

#### 2.2.3.6 setStatus()

```
bool setStatus (
            Searcher & searcher,
            ResearcherStatus value )
```

sets the status of a searcher to the given value

**Parameters**

| | |
|---|---|
| *searcher* | |
| *value* | |

**Returns**

true if success, false if not

#### 2.2.3.7 setSteps()

```
bool setSteps (
            Searcher & searcher,
            int value )
```

sets the steps of a searcher to the given value

**Parameters**

| | |
|---|---|
| *searcher* | |
| *value* | |

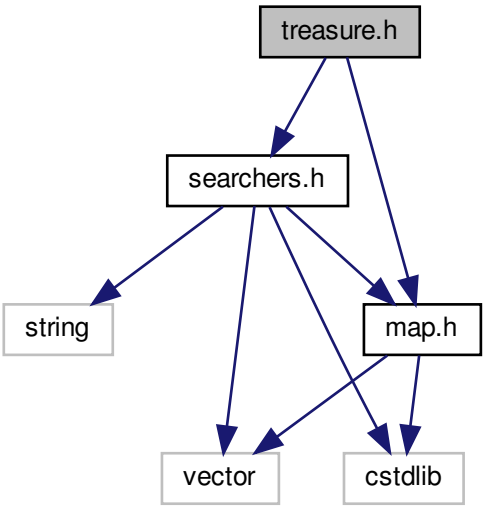
**Returns**

true if success

## 2.3 treasure.h File Reference

```
#include "map.h"
#include "searchers.h"
```

Include dependency graph for treasure.h:



## Functions

- SearcherList runSimulation (const Map &map, size_t startX, size_t startY, int nbSimulation)

  *Simulates nbSimulation searchers starting at the coordinates startX and startY on the map.*
- void runSearcher (const Map &map, size_t startX, size_t startY, Searcher &searcher)

  *walks a searcher on the map, defines the status and steps taken at the end.*
- bool getStatistics (const SearcherList &list, double &probability, double &avgSteps)

  *calculates the probability for a searcher to find the treasure, and the average steps taken to find it*

### 2.3.1 Function Documentation

#### 2.3.1.1 getStatistics()

```
bool getStatistics (
            const SearcherList & list,
            double & probability,
            double & avgSteps )
```

calculates the probability for a searcher to find the treasure, and the average steps taken to find it

**Parameters**

| | |
|---|---|
| *list* | |
| *probability* | |
| *avgSteps* | |

**Returns**

true if success, false if not

#### 2.3.1.2 runSearcher()

```
void runSearcher (
            const Map & map,
            size_t startX,
            size_t startY,
            Searcher & searcher )
```

walks a searcher on the map, defines the status and steps taken at the end.

**Parameters**

| | |
|---|---|
| *map* | |
| *startX* | |
| *startY* | |
| *searcher* | |

#### 2.3.1.3 runSimulation()

```
SearcherList runSimulation (
            const Map & map,
            size_t startX,
            size_t startY,
            int nbSimulation )
```

Simulates nbSimulation searchers starting at the coordinates startX and startY on the map.

**Parameters**

| | |
|---|---|
| *map* | |
| *startX* | |
| *startY* | |
| *nbSimulation* | |

**Returns**

a list containing the status and steps of every simulated searcher

## 2.4 utilities.h File Reference

### Functions

- int getRandomInRange (int max, int min=0)
- bool askForRestart ()

  *This function keeps asking as long as the user enters anything else than RESTART_CHAR or STOP_CHAR.*
- int askForNumberOfSimulation ()

  *This function keeps asking as long as the user enters a negative number or a char.*

### Variables

- const char RESTART_CHAR = 'Y'
- const char STOP_CHAR = 'N'

### 2.4.1 Function Documentation

#### 2.4.1.1 askForNumberOfSimulation()

```
int askForNumberOfSimulation ( )
```

This function keeps asking as long as the user enters a negative number or a char.

**Returns**

the number entered by the user

#### 2.4.1.2 askForRestart()

```
bool askForRestart ( )
```

This function keeps asking as long as the user enters anything else than RESTART_CHAR or STOP_CHAR.

**Returns**

true if the user has entered RESTART_CHAR and false if the user has entered STOP_CHAR

#### 2.4.1.3 getRandomInRange()

```
int getRandomInRange (
            int max,
            int min = 0 )
```

**Parameters**

| | |
|---|---|
| *max* | |
| *min* | |

**Returns**

a random int between min and max, both included

### 2.4.2 Variable Documentation

#### 2.4.2.1 RESTART_CHAR

```
const char RESTART_CHAR = 'Y'
```

#### 2.4.2.2 STOP_CHAR

```
const char STOP_CHAR = 'N'
```

# Index