

labo\_11\_schaufelberger\_yannick\_gallay\_david

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>File Index</b>	<b>1</b>
1.1	File List . . . . .	1
<b>2</b>	<b>File Documentation</b>	<b>3</b>
2.1	map.h File Reference . . . . .	3
2.1.1	Typedef Documentation . . . . .	4
2.1.1.1	Axe . . . . .	5
2.1.1.2	Map . . . . .	5
2.1.2	Enumeration Type Documentation . . . . .	5
2.1.2.1	MapState . . . . .	5
2.1.3	Function Documentation . . . . .	5
2.1.3.1	addLake() . . . . .	5
2.1.3.2	addRandomLake() [1/2] . . . . .	6
2.1.3.3	addRandomLake() [2/2] . . . . .	6
2.1.3.4	addRandomStart() [1/2] . . . . .	6
2.1.3.5	addRandomStart() [2/2] . . . . .	7
2.1.3.6	addRandomTreasure() . . . . .	7
2.1.3.7	addStart() . . . . .	7
2.1.3.8	addTreasure() . . . . .	8
2.1.3.9	displayWorld() . . . . .	8
2.1.3.10	getEmptyMap() . . . . .	8
2.1.3.11	getHeight() . . . . .	9
2.1.3.12	getMapValue() . . . . .	9
2.1.3.13	getWidth() . . . . .	9

2.1.3.14	<code>initWorld()</code>	10
2.1.3.15	<code>setMapValue()</code>	10
2.1.4	Variable Documentation	10
2.1.4.1	<code>NUMBER_OF_LAKE</code>	10
2.2	<code>searchers.h</code> File Reference	11
2.2.1	Typedef Documentation	12
2.2.1.1	<code>Searcher</code>	12
2.2.1.2	<code>SearcherList</code>	12
2.2.2	Enumeration Type Documentation	12
2.2.2.1	<code>ResearcherStatus</code>	12
2.2.3	Function Documentation	13
2.2.3.1	<code>displaySearcherList()</code>	13
2.2.3.2	<code>getStatus()</code>	13
2.2.3.3	<code>getStatusString()</code>	13
2.2.3.4	<code>getSteps()</code>	14
2.2.3.5	<code>initSearcher()</code>	14
2.2.3.6	<code>setStatus()</code>	14
2.2.3.7	<code>setSteps()</code>	15
2.3	<code>treasure.h</code> File Reference	15
2.3.1	Function Documentation	16
2.3.1.1	<code>getStatistics()</code>	16
2.3.1.2	<code>runSearcher()</code>	16
2.3.1.3	<code>runSimulation()</code>	17
2.4	<code>utilities.h</code> File Reference	17
2.4.1	Function Documentation	17
2.4.1.1	<code>askForNumberOfSimulation()</code>	18
2.4.1.2	<code>askForRestart()</code>	18
2.4.1.3	<code>getRandomInRange()</code>	18
2.4.2	Variable Documentation	18
2.4.2.1	<code>RESTART_CHAR</code>	18
2.4.2.2	<code>STOP_CHAR</code>	19

# Chapter 1

## File Index

### 1.1 File List

Here is a list of all files with brief descriptions:

<a href="#">map.h</a>	3
<a href="#">searchers.h</a>	11
<a href="#">treasure.h</a>	15
<a href="#">utilities.h</a>	17



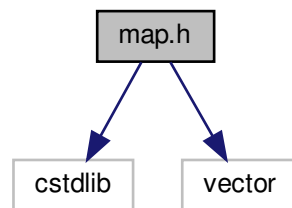
## Chapter 2

# File Documentation

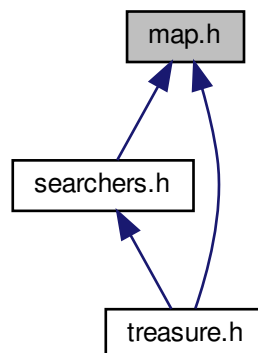
### 2.1 map.h File Reference

```
#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

<i>map</i>	
<i>originX</i>	
<i>originY</i>	
<i>radius</i>	

**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**

<i>map</i>	
------------	--

**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**

<i>map</i>	
<i>maxRadius</i>	

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

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

calls addRandomStart with an x and y parameter

**Parameters**

<i>map</i>	
------------	--

### 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

<i>map</i>	
<i>x</i>	
<i>y</i>	

### 2.1.3.6 addRandomTreasure()

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

sets a random valid cell to MS\_TREASURE

#### Parameters

<i>map</i>	
------------	--

### 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

<i>map</i>	
<i>height</i>	
<i>width</i>	

#### 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

<i>map</i>	
<i>height</i>	
<i>width</i>	

#### Returns

true if success, false if not

### 2.1.3.9 displayWorld()

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

displays the map

#### Parameters

<i>map</i>	
------------	--

### 2.1.3.10 getEmptyMap()

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

#### Parameters

<i>height</i>	
<i>width</i>	

#### Returns

an Map with the given sizes and filled with MS\_EARTH

### 2.1.3.11 getHeight()

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

#### Parameters

<i>map</i>	
------------	--

#### 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

<i>map</i>	
<i>x</i>	
<i>y</i>	

#### Returns

the MapState at the given coordinates

### 2.1.3.13 getWidth()

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

#### Parameters

<i>map</i>	
------------	--

#### 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

<i>height</i>	the height of the map
<i>width</i>	the width of the map
<i>x</i>	the x coordinate of the MS_START
<i>y</i>	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

<i>map</i>	
<i>x</i>	
<i>y</i>	
<i>value</i>	

#### Returns

true if success, false if not

## 2.1.4 Variable Documentation

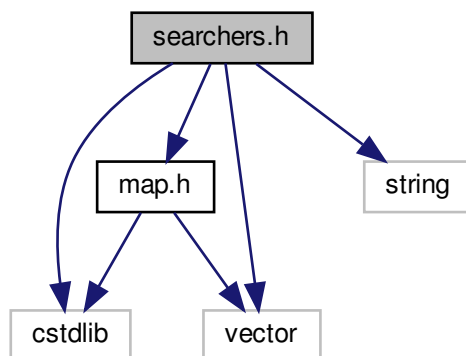
### 2.1.4.1 NUMBER\_OF\_LAKE

```
const int NUMBER_OF_LAKE = 3
```

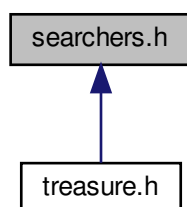
## 2.2 searchers.h File Reference

```
#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

<i>list</i>	
-------------	--

#### Returns

true if success, false if not

### 2.2.3.2 getStatus()

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

#### Parameters

<i>searcher</i>	
-----------------	--

#### Returns

the status of a searcher

### 2.2.3.3 getStatusString()

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

#### Parameters

<i>searcher</i>	
-----------------	--

**Returns**

a string containing the status of a searcher

**2.2.3.4 getSteps()**

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

**Parameters**

<i>searcher</i>	
-----------------	--

**Returns**

the amount of steps taken by a searcher

**2.2.3.5 initSearcher()**

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

**Parameters**

<i>stepsValue</i>	
<i>statusValue</i>	

**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**

<i>searcher</i>	
<i>value</i>	

**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**

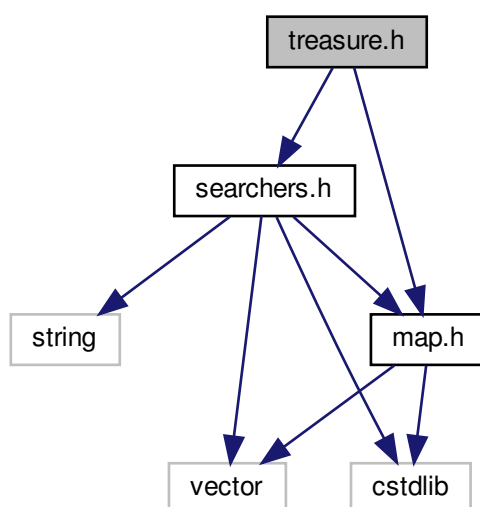
<i>searcher</i>	
<i>value</i>	

**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

<i>list</i>	
<i>probability</i>	
<i>avgSteps</i>	

##### 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

<i>map</i>	
<i>startX</i>	
<i>startY</i>	
<i>searcher</i>	

### 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

<i>map</i>	
<i>startX</i>	
<i>startY</i>	
<i>nbSimulation</i>	

#### 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

<i>max</i>	
<i>min</i>	

##### 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

- addLake
  - map.h, [5](#)
- addRandomLake
  - map.h, [6](#)
- addRandomStart
  - map.h, [6](#)
- addRandomTreasure
  - map.h, [7](#)
- addStart
  - map.h, [7](#)
- addTreasure
  - map.h, [7](#)
- askForNumberOfSimulation
  - utilities.h, [17](#)
- askForRestart
  - utilities.h, [18](#)
- Axe
  - map.h, [4](#)
- displaySearcherList
  - searchers.h, [13](#)
- displayWorld
  - map.h, [8](#)
- getEmptyMap
  - map.h, [8](#)
- getHeight
  - map.h, [8](#)
- getMapValue
  - map.h, [9](#)
- getRandomInRange
  - utilities.h, [18](#)
- getStatistics
  - treasure.h, [16](#)
- getStatus
  - searchers.h, [13](#)
- getStatusString
  - searchers.h, [13](#)
- getSteps
  - searchers.h, [14](#)
- getWidth
  - map.h, [9](#)
- initSearcher
  - searchers.h, [14](#)
- initWorld
  - map.h, [9](#)
- Map
  - map.h, [5](#)

- map.h, [3](#)
  - addLake, [5](#)
  - addRandomLake, [6](#)
  - addRandomStart, [6](#)
  - addRandomTreasure, [7](#)
  - addStart, [7](#)
  - addTreasure, [7](#)
  - Axe, [4](#)
  - displayWorld, [8](#)
  - getEmptyMap, [8](#)
  - getHeight, [8](#)
  - getMapValue, [9](#)
  - getWidth, [9](#)
  - initWorld, [9](#)
  - Map, [5](#)
  - MapState, [5](#)
  - NUMBER\_OF\_LAKE, [10](#)
  - setMapValue, [10](#)
- MapState
  - map.h, [5](#)
- NUMBER\_OF\_LAKE
  - map.h, [10](#)
- RESTART\_CHAR
  - utilities.h, [18](#)
- ResearcherStatus
  - searchers.h, [12](#)
- runSearcher
  - treasure.h, [16](#)
- runSimulation
  - treasure.h, [17](#)
- STOP\_CHAR
  - utilities.h, [18](#)
- Searcher
  - searchers.h, [12](#)
- SearcherList
  - searchers.h, [12](#)
- searchers.h, [11](#)
  - displaySearcherList, [13](#)
  - getStatus, [13](#)
  - getStatusString, [13](#)
  - getSteps, [14](#)
  - initSearcher, [14](#)
  - ResearcherStatus, [12](#)
  - Searcher, [12](#)
  - SearcherList, [12](#)
  - setStatus, [14](#)
  - setSteps, [15](#)

- setMapValue
  - map.h, [10](#)
- setStatus
  - searchers.h, [14](#)
- setSteps
  - searchers.h, [15](#)
- treasure.h, [15](#)
  - getStatistics, [16](#)
  - runSearcher, [16](#)
  - runSimulation, [17](#)
- utilities.h, [17](#)
  - askForNumberOfSimulation, [17](#)
  - askForRestart, [18](#)
  - getRandomInRange, [18](#)
  - RESTART\_CHAR, [18](#)
  - STOP\_CHAR, [18](#)