

```

1  /*
2  -----
3  Laboratory   : labo_11
4  File         : map.h
5  Author(s)    : Yannick Schaufelberger et David Gallay
6  Date        : 18.01.2020
7
8  Goal         : Library defining the Map and functions to modify, get values or
9  * display one
10 Remark(s)    :
11
12               There is the github repository:
13               https://github.com/dgheig/lab011
14
15 Compiler     : MinGW-g++ 6.3.0 and g++ 7.4.0
16 -----*/
17
18 #ifndef MAP_H
19 #define MAP_H
20
21 #include <cstdlib> //size_t
22 #include <vector>
23
24 enum MapState {
25     MS_OUT,
26     MS_EARTH,
27     MS_WATER,
28     MS_TREASURE,
29     MS_START
30 };
31
32 const int NUMBER_OF_LAKE = 3;
33
34 typedef std::vector<MapState> Axe;
35 typedef std::vector<Axe> Map;
36
37 /**
38  * @param map
39  * @return the height of the map aka the size of the vector<Axe>
40  */
41 size_t getHeight(const Map& map);
42 /**
43  * @param map
44  * @return the width of the map aka the size of the vector<MapState>
45  */
46 size_t getWidth(const Map& map);
47
48 /**
49  * @param height
50  * @param width
51  * @return an Map with the given sizes and filled with MS_EARTH
52  */
53 Map getEmptyMap(size_t height, size_t width);
54
55 /**
56  * @param map
57  * @param x
58  * @param y
59  * @return the MapState at the given coordinates
60  */
61 MapState getMapValue(const Map& map, size_t x, size_t y);
62 /**
63  * @brief sets the MapState at the given coordinates to the given value
64  * @param map
65  * @param x
66  * @param y
67  * @param value
68  * @return true if success, false if not
69  */
70 bool setMapValue(Map& map, size_t x, size_t y, MapState value);
71
72 /**
73  * @brief if the MapState at the given coordinates is MS_EARTA, sets it to MS_TREASURE
74  * @param map
75  * @param height
76  * @param width
77  * @return true if success, false if not

```

```
78  */
79  bool addTreasure(Map& map, size_t height, size_t width);
80  /**
81   * @brief sets a random valid cell to MS_TREASURE
82   * @param map
83   */
84  void addRandomTreasure(Map& map);
85
86  /**
87   * @brief if the given coordinates and radius are valid, adds a lake to the map
88   * @param map
89   * @param originX
90   * @param originY
91   * @param radius
92   * @return true if success, false if not
93   */
94  bool addLake(Map& map, size_t originX, size_t originY, size_t radius);
95  /**
96   * @brief calls addRandomLake with the same map and a radius set to the third
97   * of the smallest size
98   * @param map
99   */
100 void addRandomLake(Map& map);
101 /**
102  * @brief adds a lake at random coordinates with a random radius
103  * @param map
104  * @param maxRadius
105  */
106 void addRandomLake(Map& map, size_t maxRadius);
107
108 /**
109  * @brief if the MapState at the given coordinates is MS_EARTA, sets it to MS_START
110  * @param map
111  * @param height
112  * @param width
113  * @return true if success, false if not
114  */
115 bool addStart(Map& map, size_t x, size_t y);
116 /**
117  * @brief calls addRandomStart with an x and y parameter
118  * @param map
119  */
120 void addRandomStart(Map& map);
121 /**
122  * @brief sets a random valid cell to MS_START.
123  * @param map
124  * @param x
125  * @param y
126  */
127 void addRandomStart(Map& map, size_t& x, size_t& y);
128
129 /**
130  * @param height the height of the map
131  * @param width the width of the map
132  * @param x the x coordinate of the MS_START
133  * @param y the y coordinate of the MS_START
134  * @return a map with NUMBER_OF_LAKE lakes, one start and one treasure
135  */
136 Map initWorld(size_t height, size_t width, size_t& startX, size_t& startY);
137
138 /**
139  * @brief displays the map
140  * @param map
141  */
142 void displayWorld(const Map& map);
143
144 #endif // MAP_H
```