My Project

Generated by Doxygen 1.8.16

1 Class Index	1
1.1 Class List	1
2 Class Documentation	3
2.1 Maze Class Reference	3
2.1.1 Constructor & Destructor Documentation	3
2.1.1.1 Maze() [1/2]	3
2.1.1.2 Maze() [2/2]	4
2.1.2 Member Function Documentation	5
2.1.2.1 create()	5
2.1.2.2 get()	6
2.1.2.3 read()	6
2.1.2.4 set()	6
2.1.2.5 solve()	7
2.1.2.6 write()	7
2.1.3 Friends And Related Function Documentation	8
2.1.3.1 operator<<	8
Index	9

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:	
Maze	3

2 Class Index

Chapter 2

Class Documentation

2.1 Maze Class Reference

Public Member Functions

- Maze ()
- Maze (std::size_t rows, std::size_t cols, std::size_t begin, std::size_t end) throw (std::invalid_argument)
- bool set (const std::vector< unsigned char > &maze, std::size_t rows)
- const std::vector< unsigned char > & get () const
- bool create (std::size_t rows, std::size_t cols, std::size_t begin, std::size_t end)
- bool solve ()
- bool read (std::string filename)
- bool write (std::string filename) const

Friends

std::ostream & operator<< (std::ostream &os, const Maze &m)

2.1.1 Constructor & Destructor Documentation

2.1.1.1 Maze() [1/2]

```
Maze::Maze ( )
```

Create a blank maze of size 0-by-0

Returns

maze with rows=0 and cols=0

4 Class Documentation

2.1.1.2 Maze() [2/2]

```
Maze::Maze (
         std::size_t rows,
         std::size_t cols,
         std::size_t begin,
         std::size_t end ) throw ( std::invalid_argument)
```

Constructor will generate a random maze given input parameters. Throws exception if maze parameters imply/yield an invalid maze.

2.1 Maze Class Reference 5

Parameters

rows	-
	height
	of
	maze
cols	- width
	of
	maze
begin	- entry
	point in
	maze
end	- exit
	point
	in
	maze

2.1.2 Member Function Documentation

2.1.2.1 create()

Generate a new maze with the given parameters. Should overwrite existing maze, if any.

Parameters

rows	-
	height
	of
	maze
cols	- width
	of
	maze
begin	- entry
	point in
	maze
end	- exit
	point
	in
	maze

Returns

true if we could create a valid maze

6 Class Documentation

2.1.2.2 get()

```
const std::vector<unsigned char>& Maze::get ( ) const
```

Return the maze

Returns

an unsolved or solved maze, depending on whether it has been solved or not

2.1.2.3 read()

```
bool Maze::read (
          std::string filename )
```

Read a solved or unsolved maze from the PNG file. Should overwrite existing maze, if any

Parameters

filename	- name
	of P⇔
	NG file
	con-
	taining
	un-
	solved
	or
	solved
	maze

Returns

whether the maze could be read and is valid

2.1.2.4 set()

Function to set the maze we want to solve, validate it, and set private variables appropriately (a valid maze will only have characters listed in the project description for an unsolved maze). Should overwrite existing maze, if any.

2.1 Maze Class Reference 7

Parameters

maze	- a one
	dimen-
	sional
	vector
	repre-
	sent-
	ing an
	un-
	solved
	maze
rows	- the
	num-
	ber of
	rows
	in the
	maze

Returns

whether the maze is valid

2.1.2.5 solve()

```
bool Maze::solve ( )
```

Find the path from B to E in the current maze and record the path in the maze

Returns

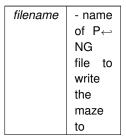
false if no path could be found from B to E or error, true otherwise

2.1.2.6 write()

Write the unsolved or solved mazed to a PNG file

8 Class Documentation

Parameters



Returns

false if the file can't be written

2.1.3 Friends And Related Function Documentation

2.1.3.1 operator <<

Operator to print maze to std::cout (useful for debugging)

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

· Maze.hpp

Index

```
create
    Maze, 5
get
    Maze, 5
Maze, 3
    create, 5
    get, 5
    Maze, 3
    operator{<<}, \color{red}{8}
    read, 6
    set, 6
    solve, 7
    write, 7
operator<<
    Maze, 8
read
    Maze, 6
set
     Maze, 6
solve
    Maze, 7
write
    Maze, 7
```