

labo_03_basset_nils_lange_yanik_gallay_david

Generated by Doxygen 1.8.13

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	Enigma Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Constructor & Destructor Documentation	5
3.1.2.1	Enigma()	5
3.1.3	Member Function Documentation	6
3.1.3.1	encrypt() [1/2]	6
3.1.3.2	encrypt() [2/2]	6
3.1.3.3	getRotors()	6
3.1.3.4	reset()	6
3.1.3.5	setReflector()	7
3.2	Reflector Class Reference	7
3.2.1	Detailed Description	7
3.2.2	Constructor & Destructor Documentation	7
3.2.2.1	Reflector()	7
3.2.3	Member Function Documentation	8
3.2.3.1	backwardTranslate()	8
3.2.3.2	getWiring()	8

3.2.3.3	translate()	8
3.3	Rotor Class Reference	9
3.3.1	Detailed Description	9
3.3.2	Constructor & Destructor Documentation	9
3.3.2.1	Rotor() [1/2]	9
3.3.2.2	Rotor() [2/2]	10
3.3.3	Member Function Documentation	10
3.3.3.1	backwardTranslate()	10
3.3.3.2	getInitialRotation()	10
3.3.3.3	getNotch()	11
3.3.3.4	getRotation()	11
3.3.3.5	getWiring()	11
3.3.3.6	reset()	11
3.3.3.7	rotate()	11
3.3.3.8	setInitialRotation() [1/2]	11
3.3.3.9	setInitialRotation() [2/2]	12
3.3.3.10	setNotch()	12
3.3.3.11	setRotation() [1/2]	12
3.3.3.12	setRotation() [2/2]	13
3.3.3.13	translate()	13
4	File Documentation	15
4.1	enigma.h File Reference	15
4.1.1	Typedef Documentation	16
4.1.1.1	Rotors	16
4.2	reflector.h File Reference	16
4.2.1	Variable Documentation	17
4.2.1.1	REFLECTOR_B	17
4.3	rotor.h File Reference	17
4.3.1	Variable Documentation	18
4.3.1.1	DEFAULT_NOTCH	18
4.3.1.2	ROTOR_I	18
4.3.1.3	ROTOR_II	18
4.3.1.4	ROTOR_III	19
4.4	utilities.h File Reference	19
4.4.1	Function Documentation	19
4.4.1.1	alphaIndex() [1/2]	19
4.4.1.2	alphaIndex() [2/2]	19
4.4.1.3	indexToChar()	20
4.4.1.4	mod()	20
	Index	21

Chapter 1

Class Index

1.1 Class List

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

Enigma	Representation of the enigma machine	5
Reflector	Class used to map uppercase letters to others	7
Rotor	Representation of a rotor of the enigma machine	9

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

enigma.h	15
reflector.h	16
rotor.h	17
utilities.h	19

Chapter 3

Class Documentation

3.1 Enigma Class Reference

Representation of the enigma machine.

```
#include <enigma.h>
```

Public Member Functions

- [Enigma](#) ([Reflector](#) reflector, const [Rotors](#) &rotors)
- char [encrypt](#) (char c)
Return the encrypted input if it is an alphabetic character, else return the character.
- std::string [encrypt](#) (const std::string &text)
Return the encrypted input.
- [Enigma](#) & [reset](#) ()
Reset the enigma machine to its initial configuration.
- [Rotors](#) & [getRotors](#) ()
get the rotors, this is the way to change them
- [Enigma](#) & [setReflector](#) (const [Reflector](#) &reflector)
set the reflector of the enigma machine

3.1.1 Detailed Description

Representation of the enigma machine.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Enigma()

```
Enigma::Enigma (
    Reflector reflector,
    const Rotors & rotors )
```

3.1.3 Member Function Documentation

3.1.3.1 `encrypt()` [1/2]

```
char Enigma::encrypt (
    char c )
```

Return the encrypted input if it is an alphabetic character, else return the character.

Parameters

<i>c</i>	character we want to encrypt
----------	------------------------------

3.1.3.2 `encrypt()` [2/2]

```
std::string Enigma::encrypt (
    const std::string & text )
```

Return the encrypted input.

Parameters

<i>text</i>	text
-------------	------

3.1.3.3 `getRotors()`

```
Rotors& Enigma::getRotors ( )
```

get the rotors, this is the way to change them

3.1.3.4 `reset()`

```
Enigma& Enigma::reset ( )
```

Reset the enigma machine to its initial configuration.

Returns

Reference on the enigma object

3.1.3.5 setReflector()

```
Enigma& Enigma::setReflector (
    const Reflector & reflector )
```

set the reflector of the enigma machine

Returns

Reference on the enigma object

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

- [enigma.h](#)

3.2 Reflector Class Reference

Class used to map uppercase letters to others.

```
#include <reflector.h>
```

Public Member Functions

- [Reflector](#) (std::string match)
- char [translate](#) (char c) const
Return the char mapped to c, it only handle uppercase characters.
- char [backwardTranslate](#) (char c) const
This is the inverse function of translate, also only handle uppercase characters.
- std::string [getWiring](#) () const

3.2.1 Detailed Description

Class used to map uppercase letters to others.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Reflector()

```
Reflector::Reflector (
    std::string match )
```

Parameters

<i>match</i>	string containing all uppercase letter once and only once, otherwise, the behaviour is undefined
--------------	--

3.2.3 Member Function Documentation

3.2.3.1 backwardTranslate()

```
char Reflector::backwardTranslate (
    char c ) const
```

This is the inverse function of translate, also only handle uppercase characters.

Parameters

<i>c</i>	character to backward translate
----------	---------------------------------

3.2.3.2 getWiring()

```
std::string Reflector::getWiring ( ) const
```

Returns

Return the wiring of the reflector

3.2.3.3 translate()

```
char Reflector::translate (
    char c ) const
```

Return the char mapped to c, it only handle uppercase characters.

Parameters

<i>c</i>	character to translate
----------	------------------------

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

- [reflector.h](#)

3.3 Rotor Class Reference

Representation of a rotor of the enigma machine.

```
#include <rotor.h>
```

Public Member Functions

- [Rotor](#) (const std::string &match, char notch=DEFAULT_NOTCH, int rotation=0)
- [Rotor](#) (const std::string &, char notch, char position)
- [Rotor](#) & [reset](#) ()
Reset the rotor to its initial states.
- char [translate](#) (char c) const
- char [backwardTranslate](#) (char c) const
This is the inverse function of translate, also only handle uppercase characters.
- bool [rotate](#) ()
Rotate the rotor.
- [Rotor](#) & [setRotation](#) (int rotation)
Set the rotation of the rotor.
- [Rotor](#) & [setRotation](#) (char rotation)
Set the rotation of the rotor.
- int [getRotation](#) () const
- [Rotor](#) & [setInitialRotation](#) (int rotation)
Set the initial rotation of the rotor.
- [Rotor](#) & [setInitialRotation](#) (char rotation)
Set the initial rotation of the rotor.
- int [getInitialRotation](#) () const
- bool [setNotch](#) (char notch)
Set the notch if it is valid.
- char [getNotch](#) () const
- std::string [getWiring](#) () const

3.3.1 Detailed Description

Representation of a rotor of the enigma machine.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Rotor() [1/2]

```
Rotor::Rotor (  
    const std::string & match,  
    char notch = DEFAULT_NOTCH,  
    int rotation = 0 )
```

Parameters

<i>match</i>	string containing all uppercase letter once and only once, otherwise, the behaviour is undefined
<i>notch</i>	notch of the rotor, if rotor steps from <code>_notch</code> to another character, the next rotor must be advanced
<i>rotation</i>	initial rotation/position of the rotor

3.3.2.2 Rotor() [2/2]

```
Rotor::Rotor (
    const std::string & ,
    char notch,
    char position )
```

Parameters

<i>match</i>	string containing all uppercase letter once and only once, otherwise, the behaviour is undefined
<i>notch</i>	notch of the rotor, if rotor steps from <code>_notch</code> to another character, the next rotor must be advanced
<i>rotation</i>	initial rotation/position of the rotor

3.3.3 Member Function Documentation**3.3.3.1 backwardTranslate()**

```
char Rotor::backwardTranslate (
    char c ) const
```

This is the inverse function of `translate`, also only handle uppercase characters.

Parameters

<i>c</i>	character to backward translate
----------	---------------------------------

3.3.3.2 getInitialRotation()

```
int Rotor::getInitialRotation ( ) const
```

Returns

The initial rotation of the rotor

3.3.3.3 getNotch()

```
char Rotor::getNotch ( ) const
```

Returns

Return the notch of the rotor

3.3.3.4 getRotation()

```
int Rotor::getRotation ( ) const
```

Returns

The current rotation of the rotor

3.3.3.5 getWiring()

```
std::string Rotor::getWiring ( ) const
```

Returns

Return the wiring of the rotor

3.3.3.6 reset()

```
Rotor& Rotor::reset ( )
```

Reset the rotor to its initial states.

Returns

Reference on the rotor

3.3.3.7 rotate()

```
bool Rotor::rotate ( )
```

Rotate the rotor.

Returns

true if the notch is passed, else return false

3.3.3.8 setInitialRotation() [1/2]

```
Rotor& Rotor::setInitialRotation (
    int rotation )
```

Set the initial rotation of the rotor.

Parameters

<i>rotation</i>	the initial rotation of the rotor
-----------------	-----------------------------------

Returns

Reference on the rotor

3.3.3.9 setInitialRotation() [2/2]

```
Rotor& Rotor::setInitialRotation (
    char rotation )
```

Set the initial rotation of the rotor.

Parameters

<i>rotation</i>	the initial rotation of the rotor
-----------------	-----------------------------------

Returns

Reference on the rotor

3.3.3.10 setNotch()

```
bool Rotor::setNotch (
    char notch )
```

Set the notch if it is valid.

Returns

true if the notch is valid, else false

3.3.3.11 setRotation() [1/2]

```
Rotor& Rotor::setRotation (
    int rotation )
```

Set the rotation of the rotor.

Parameters

<i>rotation</i>	the current rotation of the rotor
-----------------	-----------------------------------

Returns

Reference on the rotor

3.3.3.12 setRotation() [2/2]

```
Rotor& Rotor::setRotation (
    char rotation )
```

Set the rotation of the rotor.

Parameters

<i>rotation</i>	the current rotation of the rotor
-----------------	-----------------------------------

Returns

Reference on the rotor

3.3.3.13 translate()

```
char Rotor::translate (
    char c ) const
```

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

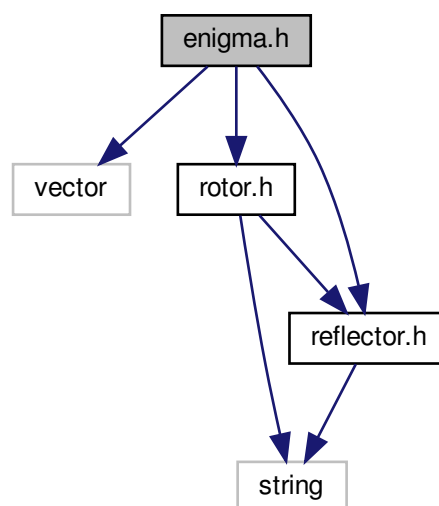
- [rotor.h](#)

Chapter 4

File Documentation

4.1 `enigma.h` File Reference

```
#include <vector>
#include "rotor.h"
#include "reflector.h"
Include dependency graph for enigma.h:
```



Classes

- class [Enigma](#)

Representation of the enigma machine.

Typedefs

- typedef std::vector< [Rotor](#) > [Rotors](#)

4.1.1 Typedef Documentation

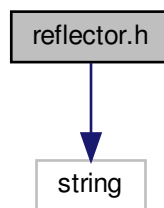
4.1.1.1 Rotors

```
typedef std::vector<Rotor> Rotors
```

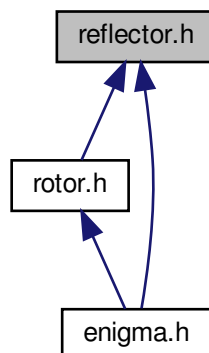
4.2 reflector.h File Reference

```
#include <string>
```

Include dependency graph for reflector.h:



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



Classes

- class [Reflector](#)

Class used to map uppercase letters to others.

Variables

- const [Reflector](#) REFLECTOR_B

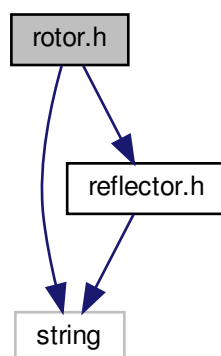
4.2.1 Variable Documentation

4.2.1.1 REFLECTOR_B

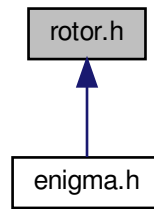
```
const Reflector REFLECTOR_B
```

4.3 rotor.h File Reference

```
#include <string>
#include "reflector.h"
Include dependency graph for rotor.h:
```



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



Classes

- class [Rotor](#)
Representation of a rotor of the enigma machine.

Variables

- const char [DEFAULT_NOTCH](#) = 'A'
- const [Rotor](#) [ROTOR_I](#)
- const [Rotor](#) [ROTOR_II](#)
- const [Rotor](#) [ROTOR_III](#)

4.3.1 Variable Documentation

4.3.1.1 DEFAULT_NOTCH

```
const char DEFAULT_NOTCH = 'A'
```

4.3.1.2 ROTOR_I

```
const Rotor ROTOR_I
```

4.3.1.3 ROTOR_II

```
const Rotor ROTOR_II
```

4.3.1.4 ROTOR_III

```
const Rotor ROTOR_III
```

4.4 utilities.h File Reference

Functions

- char `indexToChar` (int index)
- unsigned `mod` (int a, int b)
since operator% is not mathematical modulo but reminder operator, this version ensure a positive result
- unsigned `alphaIndex` (int index)
- int `alphaIndex` (char c)

4.4.1 Function Documentation

4.4.1.1 `alphaIndex()` [1/2]

```
unsigned alphaIndex (
    int index )
```

Parameters

<i>a</i>	index that may be invalid
----------	---------------------------

Returns

return the valid index corresponding to a

4.4.1.2 `alphaIndex()` [2/2]

```
int alphaIndex (
    char c )
```

Parameters

<i>c</i>	char of which we want to have the corresponding index
----------	---

Returns

return the valid index corresponding to c

4.4.1.3 indexToChar()

```
char indexToChar (
    int index )
```

Parameters

<i>index</i>	index of a char in the alphabete
--------------	----------------------------------

Returns

char corresponding to index

4.4.1.4 mod()

```
unsigned mod (
    int a,
    int b )
```

since operator% is not mathematical modulo but reminder operator, this version ensure a positive result

Parameters

<i>a</i>	left operand
<i>b</i>	right operand

Returns

return the modulo of a and b

Index

- alphaIndex
 - utilities.h, 19
- backwardTranslate
 - Reflector, 8
 - Rotor, 10
- DEFAULT_NOTCH
 - rotor.h, 18
- encrypt
 - Enigma, 6
- Enigma, 5
 - encrypt, 6
 - Enigma, 5
 - getRotors, 6
 - reset, 6
 - setReflector, 6
- enigma.h, 15
 - Rotors, 16
- getInitialRotation
 - Rotor, 10
- getNotch
 - Rotor, 10
- getRotation
 - Rotor, 11
- getRotors
 - Enigma, 6
- getWiring
 - Reflector, 8
 - Rotor, 11
- indexToChar
 - utilities.h, 20
- mod
 - utilities.h, 20
- REFLECTOR_B
 - reflector.h, 17
- ROTOR_III
 - rotor.h, 18
- ROTOR_II
 - rotor.h, 18
- ROTOR_I
 - rotor.h, 18
- Reflector, 7
 - backwardTranslate, 8
 - getWiring, 8
 - Reflector, 7
 - translate, 8
- reflector.h, 16
 - REFLECTOR_B, 17
- reset
 - Enigma, 6
 - Rotor, 11
- rotate
 - Rotor, 11
- Rotor, 9
 - backwardTranslate, 10
 - getInitialRotation, 10
 - getNotch, 10
 - getRotation, 11
 - getWiring, 11
 - reset, 11
 - rotate, 11
 - Rotor, 9, 10
 - setInitialRotation, 11, 12
 - setNotch, 12
 - setRotation, 12, 13
 - translate, 13
- rotor.h, 17
 - DEFAULT_NOTCH, 18
 - ROTOR_III, 18
 - ROTOR_II, 18
 - ROTOR_I, 18
- Rotors
 - enigma.h, 16
- setInitialRotation
 - Rotor, 11, 12
- setNotch
 - Rotor, 12
- setReflector
 - Enigma, 6
- setRotation
 - Rotor, 12, 13
- translate
 - Reflector, 8
 - Rotor, 13
- utilities.h, 19
 - alphaIndex, 19
 - indexToChar, 20
 - mod, 20