

labo\_03\_basset\_nils\_lange\_yanik\_gallay\_david

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
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 . . . . .	7
3.2.3.1	backwardTranslate() . . . . .	7
3.2.3.2	translate() . . . . .	8

3.3	Rotor Class Reference	8
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]	9
3.3.3	Member Function Documentation	9
3.3.3.1	backwardTranslate()	9
3.3.3.2	getNotch()	9
3.3.3.3	getRotation()	10
3.3.3.4	reset()	10
3.3.3.5	rotate()	10
3.3.3.6	setNotch()	10
3.3.3.7	setRotation() [1/2]	10
3.3.3.8	setRotation() [2/2]	11
3.3.3.9	translate()	11
4	File Documentation	13
4.1	enigma.h File Reference	13
4.1.1	Typedef Documentation	14
4.1.1.1	Rotors	14
4.2	reflector.h File Reference	14
4.2.1	Variable Documentation	15
4.2.1.1	reflector_B	15
4.3	rotor.h File Reference	15
4.3.1	Variable Documentation	16
4.3.1.1	DEFAULT_NOTCH	16
4.3.1.2	rotor_I	16
4.3.1.3	rotor_II	16
4.3.1.4	rotor_III	17
4.4	utilities.h File Reference	17
4.4.1	Function Documentation	17
4.4.1.1	alphaIndex() [1/2]	17
4.4.1.2	alphaIndex() [2/2]	17
4.4.1.3	indexToChar()	18
4.4.1.4	mod()	18
	Index	19

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Enigma</a>	Representation of the enigma machine . . . . .	<a href="#">5</a>
<a href="#">Reflector</a>	Class used to map uppercase letters to others . . . . .	<a href="#">7</a>
<a href="#">Rotor</a>	Representation of a rotor of the enigma machine . . . . .	<a href="#">8</a>



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">enigma.h</a>	13
<a href="#">reflector.h</a>	14
<a href="#">rotor.h</a>	15
<a href="#">utilities.h</a>	17





## Chapter 3

# Class Documentation

### 3.1 Enigma Class Reference

Representation of the enigma machine.

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

#### Public Member Functions

- [Enigma](#) ([Reflector](#) reflector, std::vector< [Rotor](#) > 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,
    std::vector< Rotor > 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.*

### 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 )
```

### 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

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

3.2.3.2 `translate()`

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

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

## Parameters

<code>c</code>	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](#) (std::string match, char notch=[DEFAULT\\_NOTCH](#), int rotation=0)
- [Rotor](#) (std::string match, 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
- bool [setNotch](#) (char notch)  
*Set the notch if it is valid.*
- char [getNotch](#) () 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 (
    std::string match,
    char notch = DEFAULT_NOTCH,
    int rotation = 0 )
```

#### 3.3.2.2 Rotor() [2/2]

```
Rotor::Rotor (
    std::string match,
    char notch,
    char position )
```

### 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 getNotch()

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

**Returns**

Return the notch of the rotor

**3.3.3.3 getRotation()**

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

**Returns**

The current rotation of the rotor

**3.3.3.4 reset()**

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

Reset the rotor to its initial states.

**Returns**

Reference on the rotor

**3.3.3.5 rotate()**

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

Rotate the rotor.

**Returns**

true if the notch is passed, else return false

**3.3.3.6 setNotch()**

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

Set the notch if it is valid.

**Returns**

true if the notch is valid, else false

**3.3.3.7 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

**3.3.3.8** `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

**3.3.3.9** `translate()`

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

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

- [rotor.h](#)



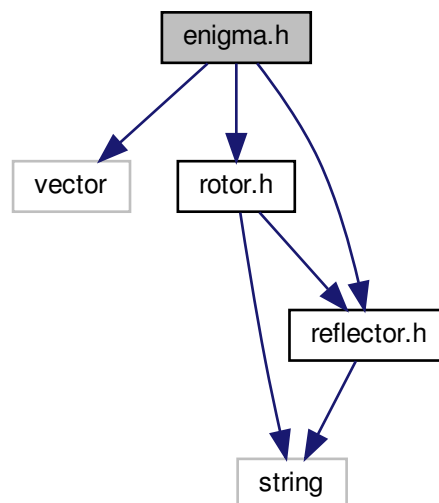


## Chapter 4

# 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

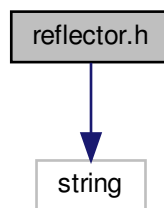
#### 4.1.1.1 Rotors

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

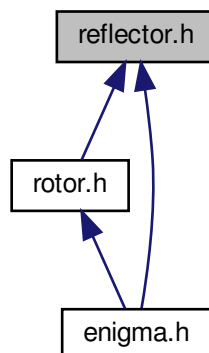
## 4.2 reflector.h File Reference

```
#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`

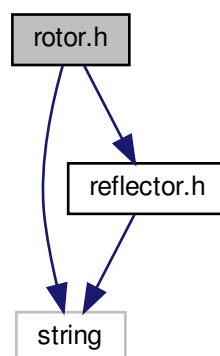
### 4.2.1 Variable Documentation

#### 4.2.1.1 reflector\_B

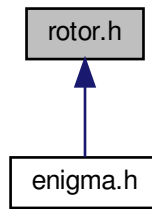
```
const Reflector reflector_B
```

## 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, 17

backwardTranslate  
    Reflector, 7  
    Rotor, 9

DEFAULT\_NOTCH  
    rotor.h, 16

encrypt  
    Enigma, 6  
Enigma, 5  
    encrypt, 6  
    Enigma, 5  
    getRotors, 6  
    reset, 6  
    setReflector, 6  
enigma.h, 13  
    Rotors, 14

getNotch  
    Rotor, 9  
getRotation  
    Rotor, 10  
getRotors  
    Enigma, 6

indexToChar  
    utilities.h, 18

mod  
    utilities.h, 18

Reflector, 7  
    backwardTranslate, 7  
    Reflector, 7  
    translate, 8

reflector.h, 14  
    reflector\_B, 15

reflector\_B  
    reflector.h, 15

reset  
    Enigma, 6  
    Rotor, 10

rotate  
    Rotor, 10

Rotor, 8  
    backwardTranslate, 9  
    getNotch, 9  
    getRotation, 10

reset, 10  
rotate, 10  
Rotor, 9  
setNotch, 10  
setRotation, 10, 11  
translate, 11  
rotor.h, 15  
    DEFAULT\_NOTCH, 16  
    rotor\_III, 16  
    rotor\_II, 16  
    rotor\_I, 16  
rotor\_III  
    rotor.h, 16  
rotor\_II  
    rotor.h, 16  
rotor\_I  
    rotor.h, 16  
Rotors  
    enigma.h, 14

setNotch  
    Rotor, 10  
setReflector  
    Enigma, 6  
setRotation  
    Rotor, 10, 11

translate  
    Reflector, 8  
    Rotor, 11

utilities.h, 17  
    alphaIndex, 17  
    indexToChar, 18  
    mod, 18