

# Шифр Гронсвельда

Создано системой Doxygen 1.9.4



1 Иерархический список классов	1
1.1 Иерархия классов . . . . .	1
2 Алфавитный указатель классов	3
2.1 Классы . . . . .	3
3 Список файлов	5
3.1 Файлы . . . . .	5
4 Классы	7
4.1 Класс cipher_error . . . . .	7
4.1.1 Подробное описание . . . . .	8
4.1.2 Конструктор(ы) . . . . .	8
4.1.2.1 cipher_error() [1/2] . . . . .	8
4.1.2.2 cipher_error() [2/2] . . . . .	8
4.2 Структура KeyB_fixture . . . . .	9
4.3 Класс modAlphaCipher . . . . .	9
4.3.1 Подробное описание . . . . .	10
4.3.2 Конструктор(ы) . . . . .	10
4.3.2.1 modAlphaCipher() . . . . .	10
4.3.3 Методы . . . . .	11
4.3.3.1 decrypt() . . . . .	11
4.3.3.2 encrypt() . . . . .	11
4.3.3.3 getValidCipherText() . . . . .	12
4.3.3.4 getValidKey() . . . . .	12
4.3.3.5 getValidOpenText() . . . . .	12
5 Файлы	15
5.1 Файл modAlphaCipher.h . . . . .	15
5.1.1 Подробное описание . . . . .	15
5.2 modAlphaCipher.h . . . . .	16
Предметный указатель	17



# Глава 1

## Иерархический список классов

### 1.1 Иерархия классов

Иерархия классов.

std::invalid_argument	
cipher_error . . . . .	7
KeyB_fixture . . . . .	9
modAlphaCipher . . . . .	9



## Глава 2

# Алфавитный указатель классов

### 2.1 Классы

Классы с их кратким описанием.

<a href="#">cipher_error</a>		
	Обработка исключений . . . . .	7
<a href="#">KeyB_fixture</a>	. . . . .	9
<a href="#">modAlphaCipher</a>		
	Шифрование методом Гронсфельда . . . . .	9





## Глава 3

# Список файлов

### 3.1 Файлы

Полный список документированных файлов.

<a href="#">modAlphaCipher.h</a>	15
----------------------------------	----



## Глава 4

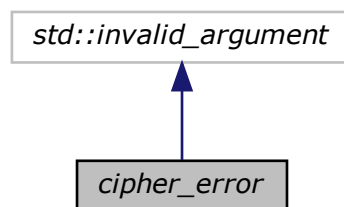
# Классы

### 4.1 Класс `cipher_error`

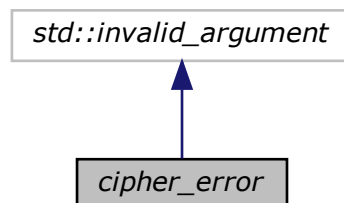
Обработка исключений

```
#include <modAlphaCipher.h>
```

Граф наследования: `cipher_error`:



Граф связей класса `cipher_error`:



## Открытые члены

- [cipher\\_error](#) (const std::string &what\_arg)  
Конструктор с аргументом типа std::string.
- [cipher\\_error](#) (const char \*what\_arg)  
Конструктор с аргументом типа const char\*.

### 4.1.1 Подробное описание

#### Обработка исключений

Класс, созданный для обработки ошибок

### 4.1.2 Конструктор(ы)

#### 4.1.2.1 cipher\_error() [1/2]

```
cipher_error::cipher_error (
    const std::string & what_arg )  [inline], [explicit]
```

Конструктор с аргументом типа std::string.

#### Аргументы

what_arg	Сообщение об ошибке.
----------	----------------------

#### 4.1.2.2 cipher\_error() [2/2]

```
cipher_error::cipher_error (
    const char * what_arg )  [inline], [explicit]
```

Конструктор с аргументом типа const char\*.

#### Аргументы

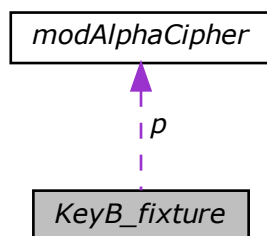
what_arg	Сообщение об ошибке.
----------	----------------------

Объявления и описания членов класса находятся в файле:

- [modAlphaCipher.h](#)

## 4.2 Структура KeyB\_fixture

Граф связей класса KeyB\_fixture:



Открытые атрибуты

- `modAlphaCipher * p`

Объявления и описания членов структуры находятся в файле:

- `main.cpp`

## 4.3 Класс modAlphaCipher

Шифрование методом Гронсфелда

```
#include <modAlphaCipher.h>
```

Открытые члены

- `modAlphaCipher ()=delete`  
Запрет на использование конструктора по умолчанию
- `modAlphaCipher (const std::wstring &skey)`  
Конструктор для установки ключа
- `std::wstring encrypt (const std::wstring &open_text)`  
Зашифровывание
- `std::wstring decrypt (const std::wstring &cipher_text)`  
Расшифровывание

## Закрытые члены

- `std::vector< int > convert (const std::wstring &s)`  
Преобразование строки в вектор
- `std::wstring convert (const std::vector< int > &v)`  
Преобразование вектора в строку
- `std::wstring getValidKey (const std::wstring &s)`  
Валидация ключа
- `std::wstring getValidOpenText (const std::wstring &s)`  
Валидация открытого текста
- `std::wstring getValidCipherText (const std::wstring &s)`  
Валидация зашифрованного текста

## Закрытые данные

- `std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"`  
Алфавит
- `std::map< char, int > alphaNum`  
Ассоциативный массив "номер по символу".
- `std::vector< int > key`  
Ключ

### 4.3.1 Подробное описание

#### Шифрование методом Гронсфелда

Ключ устанавливается в конструкторе. Для зашифровывания и расшифровывания предназначены методы `encrypt` и `decrypt`.

#### Предупреждения

Реализация только для русского языка

### 4.3.2 Конструктор(ы)

#### 4.3.2.1 `modAlphaCipher()`

```
modAlphaCipher::modAlphaCipher (
    const std::wstring & skey )
```

Конструктор для установки ключа

#### Аргументы

<code>skey</code>	Строка, представляющая ключ для шифрования.
-------------------	---

### 4.3.3 Методы

#### 4.3.3.1 decrypt()

```
std::wstring modAlphaCipher::decrypt (
    const std::wstring & cipher_text )
```

Расшифровывание

Аргументы

in	cipher_text	Шифрованный текст должен быть пустой строкой. Строчные символы автоматически преобразуются к прописным. Все не-буквы удаляются
----	-------------	--

Возвращает

Зашифрованная строка

Исключения

<a href="#">cipher_error</a> ,если	текст пустой
------------------------------------	--------------

#### 4.3.3.2 encrypt()

```
std::wstring modAlphaCipher::encrypt (
    const std::wstring & open_text )
```

Зашифровывание

Аргументы

in	open_text	Открытый текст. Не должен быть пустой строкой. Строчные символы автоматически преобразуются к прописным. Все не-буквы удаляются
----	-----------	---

Возвращает

Зашифрованная строка

Исключения

<a href="#">cipher_error</a> ,если	текст пустой
------------------------------------	--------------

#### 4.3.3.3 `getValidCipherText()`

```
std::wstring modAlphaCipher::getValidCipherText (  
    const std::wstring & s ) [inline], [private]
```

Валидация зашифрованного текста

Проверяет, что зашифрованный текст содержит только допустимые символы алфавита.

Аргументы

ws	Входная строка, представляющая зашифрованный текст.
----	---

Исключения

<a href="#">cipher_error</a>	Если зашифрованный текст содержит недопустимые символы.
------------------------------	---

#### 4.3.3.4 `getValidKey()`

```
std::wstring modAlphaCipher::getValidKey (  
    const std::wstring & s ) [private]
```

Валидация ключа

Проверяет, что ключ не пустой и не содержит символов, не принадлежащих алфавиту.

Аргументы

s	Входная строка, представляющая ключ.
---	--------------------------------------

Исключения

<a href="#">cipher_error</a>	Если ключ пустой или содержит недопустимые символы.
------------------------------	---

#### 4.3.3.5 `getValidOpenText()`

```
std::wstring modAlphaCipher::getValidOpenText (  
    const std::wstring & s ) [inline], [private]
```

Валидация открытого текста

Проверяет, что открытый текст содержит только допустимые символы алфавита.



## Аргументы

ws	Входная строка, представляющая открытый текст.
----	--

## Исключения

<a href="#">cipher_error</a>	Если открытый текст содержит недопустимые символы.
------------------------------	--

Объявления и описания членов классов находятся в файлах:

- [modAlphaCipher.h](#)
- [modAlphaCipher.cpp](#)



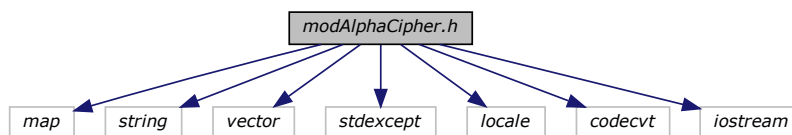
## Глава 5

# Файлы

### 5.1 Файл modAlphaCipher.h

```
#include <map>
#include <string>
#include <vector>
#include <stdexcept>
#include <locale>
#include <codecvt>
#include <iostream>
```

Граф включаемых заголовочных файлов для modAlphaCipher.h:



### Классы

- class [modAlphaCipher](#)  
Шифрование методом Гронсфельда
- class [cipher\\_error](#)  
Обработка исключений

#### 5.1.1 Подробное описание

#### Автор

Маштаков Д.С.

Версия

1.0

Дата

22.11.2024

Авторство

ИБСТ ПГУ

## 5.2 modAlphaCipher.h

[См. документацию.](#)

```
1
8 #pragma once
9 #include <map>
10 #include <string>
11 #include <vector>
12 #include <stdexcept>
13 #include <locale>
14 #include <codecvt>
15 #include <iostream>
16
22 class modAlphaCipher
23 {
24 private:
25     std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";
26     std::map<char, int> alphaNum;
27     std::vector<int> key;
28     std::vector<int> convert(const std::wstring& s);
29     std::wstring convert(const std::vector<int>& v);
30     std::wstring getValidKey(const std::wstring & s);
47     std::wstring getValidOpenText(const std::wstring& s);
56     std::wstring getValidCipherText(const std::wstring& s);
57
58 public:
59     modAlphaCipher() = delete;
65     modAlphaCipher(const std::wstring& skey);
74     std::wstring encrypt(const std::wstring& open_text);
83     std::wstring decrypt(const std::wstring& cipher_text);
84 };
91 class cipher_error : public std::invalid_argument
92 {
93 public:
99     explicit cipher_error(const std::string& what_arg):
100         std::invalid_argument(what_arg){}
106     explicit cipher_error(const char* what_arg):
107         std::invalid_argument(what_arg){}
108 };
```

# Предметный указатель

- cipher\_error, [7](#)
  - cipher\_error, [8](#)
- decrypt
  - modAlphaCipher, [11](#)
- encrypt
  - modAlphaCipher, [11](#)
- getValidCipherText
  - modAlphaCipher, [11](#)
- getValidKey
  - modAlphaCipher, [12](#)
- getValidOpenText
  - modAlphaCipher, [12](#)
- KeyB\_fixture, [9](#)
- modAlphaCipher, [9](#)
  - decrypt, [11](#)
  - encrypt, [11](#)
  - getValidCipherText, [11](#)
  - getValidKey, [12](#)
  - getValidOpenText, [12](#)
  - modAlphaCipher, [10](#)
- modAlphaCipher.h, [15](#)