

# Шифр методом Гронсфельда

## 1.0

Создано системой Doxygen 1.9.1



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 modAlpha.h	15
5.1.1 Подробное описание	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 Класс,созданный для обработки ошибок . . . . .	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="#">modAlpha.h</a>	
Заголовочный файл для модуля Гронсфельда . . . . .	<a href="#">15</a>



## Глава 4

# Классы

### 4.1 Класс cipher\_error

Обработка исключений Класс, созданный для обработки ошибок

```
#include <modAlpha.h>
```

Граф наследования: cipher\_error:



Граф связей класса cipher\_error:



## Открытые члены

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

### 4.1.1 Подробное описание

Обработка исключений Класс, созданный для обработки ошибок

Наследуется от std::invalid\_argument

### 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 параметром типа const std::string.

Аргументы

what_arg	Описание ошибки
----------	-----------------

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

```

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

```

Конструктор с параметром типа const char.

Аргументы

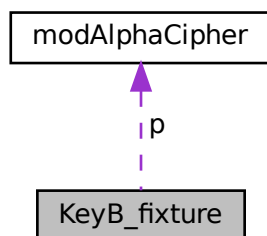
what_arg	Описание ошибки
----------	-----------------

Объявления и описания членов класса находятся в файле:

- [modAlpha.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 Гронсфельда

```
#include <modAlpha.h>
```

Открытые члены

- `modAlphaCipher ()=delete`  
Запрет на использование конструктора по умолчанию
- `modAlphaCipher (const std::wstring &skey)`  
Конструктор с параметром
- `std::wstring encrypt (const std::wstring &open_text)`  
Метод для зашифровывания
- `std::wstring decrypt (const std::wstring &cipher_text)`  
Метод для расшифровывания

## 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 по порядку
- `std::map< wchar_t, int > alphaNum`  
Ассоциативный массив "номер по символу".
- `std::vector< int > key`  
Ключ

### 4.3.1 Подробное описание

#### Шифрование методом Гронсфельда

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 с параметром

Аргументы

skey	Ключ для шифрования/расшифрования
------	-----------------------------------

## Исключения

<a href="#">cipher_error</a>	Если ключ невалидный
------------------------------	----------------------

## 4.3.3 Методы

## 4.3.3.1 decrypt()

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

Метод для расшифрования

## Аргументы

<code>cipher_text</code>	Зашифрованный текст
--------------------------	---------------------

## Исключения

<a href="#">cipher_error</a>	Если зашифрованный текст невалидный
------------------------------	-------------------------------------

## Возвращает

Расшифрованный текст

## 4.3.3.2 encrypt()

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

Метод для зашифровывания

## Аргументы

<code>open_text</code>	Открытый текст Строчные символы автоматически преобразуются к прописным. Все символы, не являющиеся буквами, удаляются
------------------------	--

## Исключения

<a href="#">cipher_error</a>	Если открытый текст невалидный
------------------------------	--------------------------------

Возвращает

Зашифрованный текст

#### 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

s	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

s	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:

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



## Глава 5

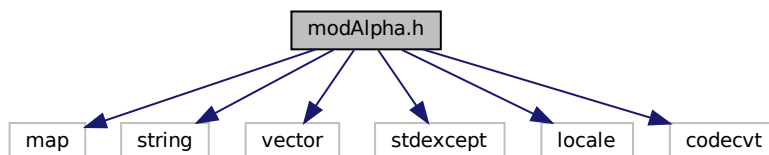
# Файлы

### 5.1 Файл modAlpha.h

Заголовочный файл для модуля Гронсфельда

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

Граф включаемых заголовочных файлов для modAlpha.h:



## Классы

- class `modAlphaCipher`  
Шифрование методом Гронсфельда
- class `cipher_error`  
Обработка исключений Класс, созданный для обработки ошибок

### 5.1.1 Подробное описание

Заголовочный файл для модуля Гронсфельда

Автор

Синельникова Т.А.

Версия

1.0

Дата

01.12.2024

Авторство

ИБСТ ПГУ

Предупреждения

Это учебный пример

# Предметный указатель

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