

# Шифрование методом Гронсвельда

## 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
4.2 Структура KeyAB_fixture . . . . .	8
4.3 Класс modAlphaCipher . . . . .	8
4.3.1 Подробное описание . . . . .	9
4.3.2 Конструктор(ы) . . . . .	9
4.3.2.1 modAlphaCipher() . . . . .	9
4.3.3 Методы . . . . .	10
4.3.3.1 convert() [1/2] . . . . .	10
4.3.3.2 convert() [2/2] . . . . .	10
4.3.3.3 decrypt() . . . . .	11
4.3.3.4 encrypt() . . . . .	12
4.3.3.5 getValidKey() . . . . .	13
4.3.3.6 getValidText() . . . . .	13
5 Файлы	15
5.1 Файл modAlphaCipher.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
KeyAB_fixture . . . . .	8
modAlphaCipher . . . . .	8

## Глава 2

# Алфавитный указатель классов

### 2.1 Классы

Классы с их кратким описанием.

<a href="#">cipher_error</a>	Класс, предназначенный для обработки исключений . . . . .	7
<a href="#">KeyAB_fixture</a>	. . . . .	8
<a href="#">modAlphaCipher</a>	Класс, который реализует шифрование методом "Гронсвельда" . . . . .	8

## Глава 3

# Список файлов

### 3.1 Файлы

Полный список документированных файлов.

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

## Глава 4

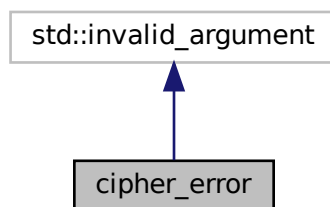
# Классы

### 4.1 Класс `cipher_error`

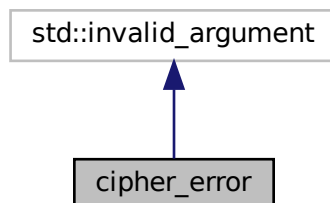
Класс, предназначенный для обработки исключений

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

Граф наследования: `cipher_error`:



Граф связей класса `cipher_error`:



## Открытые члены

- `cipher_error (const std::string &what_arg)`
- `cipher_error (const char *what_arg)`

### 4.1.1 Подробное описание

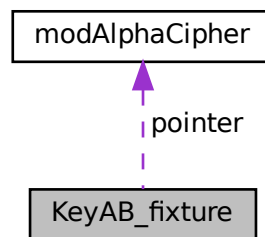
Класс, предназначенный для обработки исключений

Объявления и описания членов класса находятся в файле:

- [modAlphaCipher.h](#)

## 4.2 Структура KeyAB\_fixture

Граф связей класса KeyAB\_fixture:



## Открытые атрибуты

- [modAlphaCipher](#) \* pointer

Объявления и описания членов структуры находятся в файле:

- `main.cpp`

## 4.3 Класс modAlphaCipher

Класс, который реализует шифрование методом "Гронсвельда".

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

## Открытые члены

- `modAlphaCipher ()=delete`  
Запрещённый конструктор без параметров
- `modAlphaCipher (const std::wstring &skey)`  
Конструктор для ключа
- `std::wstring encrypt (const std::wstring &open_text)`  
Метод для шифрования
- `std::wstring decrypt (const std::wstring &cipher_text)`  
Метод, предназначенный для расшифрования

## 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 getValidText (const std::wstring &s)`  
Валидация текста при шифровании или расшифровании

## Закрытые данные

- `std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"`  
Используемый алфавит по порядку для сообщений, которые шифруются методом "Гронсвельда".
- `std::map< char, int > alphaNum`  
Ассоциативный массив "номер по символу".
- `std::vector< int > key`  
Атрибут, хранящий в себе ключ для шифрования и расшифрования

## 4.3.1 Подробное описание

Класс, который реализует шифрование методом "Гронсвельда".

## Предупреждения

Работает только с русскоязычными сообщениями

## 4.3.2 Конструктор(ы)

## 4.3.2.1 modAlphaCipher()

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

## Конструктор для ключа

Цикл `for` построен по строке-алфавиту и на каждом шаге добавляет в ассоциативный массив символ и его номер.

```
for (unsigned i=0; i<numAlpha.size(); i++) {
    alphaNum[numAlpha[i]]=i;
}
```



## Аргументы

<code>std::wstring</code>	- ключ в виде строки
---------------------------	----------------------

## 4.3.3 Методы

4.3.3.1 `convert()` [1/2]

```
std::wstring modAlphaCipher::convert (
    const std::vector< int > & v ) [inline], [private]
```

## Преобразование вектора в строку

В переменную типа "wstring" с именем "result" записывается строка согласно индексам каждой буквы алфавита "numAlpha". Индексы хранятся в векторе типа "int", который поступил на вход.

```
wstring result;
for(auto i:v) {
    result.push_back(numAlpha[i]);
}
```

## Возвращает

строка текста типа "wstring"

4.3.3.2 `convert()` [2/2]

```
std::vector< int > modAlphaCipher::convert (
    const std::wstring & s ) [inline], [private]
```

## Преобразование строки в вектор

В вектор типа "int" с именем "result" записываются числа, которые являются индексами алфавита "numAlpha", применяемый для строки, которая поступила на вход.

```
vector<int> result;
for(auto c:s) {
    result.push_back(alphaNum[c]);
}
```

## Возвращает

`std::vector <int>`, в котором хранятся индексы букв сообщения из алфавита "numAlpha"

#### 4.3.3.3 decrypt()

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

Метод, предназначенный для расшифрования

Здесь сначала формируется вектор `work` из строки шифротекста с помощью метода `convert()`. А также происходит проверка шифротекста на наличие ошибки при помощи метода `getValidAlphabetText()`.

```
vector<int> work = convert(getValidAlphabetText(cipher_text));
```

Если при зашифровании мы прибавляли значение ключа, то при расшифровании значения ключа надо вычитать. А чтобы не получить отрицательных значений, выполняется еще прибавление значения модуля, так как такое прибавление не влияет на результат модулю.

```
for(unsigned i=0; i < work.size(); i++) {
    work[i] = (work[i] + alphaNum.size() - key[i % key.size()]) % alphaNum.size();
}
```

## Аргументы

<code>std::wstring</code>	<code>cipher_text</code> - сообщение, которое нужно расшифровать
---------------------------	------------------------------------------------------------------

## Исключения

<code>cipher_error</code> , если	строка, которая поступила на вход пустая или в ней есть недопустимые символы
----------------------------------	------------------------------------------------------------------------------

## Возвращает

строка расшифрованного текста типа `"wstring"`

4.3.3.4 `encrypt()`

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

## Метод для шифрования

Здесь сначала формируется вектор `work` из строки открытого текста с помощью метода `convert()`. А также происходит проверка текста на наличие ошибки при помощи метода `getValidAlphabetText()`.

```
vector<int> work = convert(getValidAlphabetText(open_text));
```

Затем в цикле к каждому элементу вектора прибавляется элемент ключа по модулю размера алфавита. Так как ключ может быть короче текста, то при индексации ключа выполняется операция по модулю размера ключа. Это позволяет использовать ключ циклически на длинных сообщениях.

```
for(unsigned i=0; i < work.size(); i++) {
    work[i] = (work[i] + key[i % key.size()]) % alphaNum.size();
}
```

Далее, при возврате значения, вектор `work` опять преобразуется в строку.

## Аргументы

<code>std::wstring</code>	<code>open_text</code> - сообщение, которое нужно зашифровать
---------------------------	---------------------------------------------------------------

## Исключения

<code>cipher_error</code> , если	строка, которая поступила на вход пустая или в ней есть недопустимые символы
----------------------------------	------------------------------------------------------------------------------

## Возвращает

строка зашифрованного текста типа `"wstring"`

## 4.3.3.5 getValidKey()

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

## Валидация ключа

Сначала введённый ключ проверяется на пустоту при помощи обычного условия. Если ключ не пустой, то он проверяется на наличие недопустимых символов.

## Предупреждения

Строчные буквы алфавита переводятся в прописные.

## Аргументы

std::wstring	s - ключ, который нужно проверить на наличие ошибок, в виде строки
--------------	--------------------------------------------------------------------

## Исключения

<a href="#">cipher_error</a> , если	ключ пустой или в нём присутствуют недопустимые символы
-------------------------------------	---------------------------------------------------------

## Возвращает

Ключ в виде строки типа "wstring", который успешно прошёл валидацию

## 4.3.3.6 getValidText()

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

## Валидация текста при шифровании или расшифровании

Сначала введённый текст проверяется на пустоту при помощи обычного условия. Если текст не пустой, то он проверяется на наличие недопустимых символов.

## Предупреждения

Строчные буквы алфавита переводятся в прописные.

## Аргументы

std::wstring	s - строка текста для шифрования или расшифрования, которая проверяется на наличие ошибок
--------------	-------------------------------------------------------------------------------------------

## Исключения

<a href="#">cipher_error</a> , если	текст является пустым или в нём присутствуют недопустимые символы
-------------------------------------	-------------------------------------------------------------------

## Возвращает

Текст в виде строки типа "wstring", который успешно прошёл валидацию

Объявления и описания членов классов находятся в файлах:

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

## Глава 5

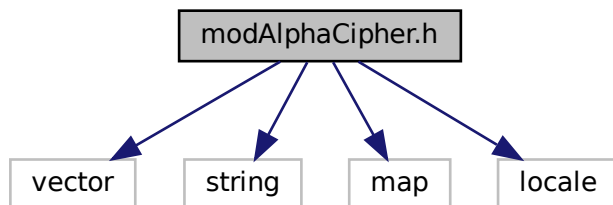
# Файлы

### 5.1 Файл modAlphaCipher.h

Описание класса `modAlphaCipher`.

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

Граф включаемых заголовочных файлов для modAlphaCipher.h:



### Классы

- class `modAlphaCipher`  
Класс, который реализует шифрование методом "Гронсвельда".
- class `cipher_error`  
Класс, предназначенный для обработки исключений

### 5.1.1 Подробное описание

Описание класса [modAlphaCipher](#).

Автор

Пицан А. П.

Версия

1.0

Дата

7.11.2022

Авторство

ИБСТ ПГУ

# Предметный указатель

- cipher\_error, [7](#)
- convert
  - modAlphaCipher, [10](#)
- decrypt
  - modAlphaCipher, [10](#)
- encrypt
  - modAlphaCipher, [12](#)
- getValidKey
  - modAlphaCipher, [12](#)
- getValidText
  - modAlphaCipher, [13](#)
- KeyAB\_fixture, [8](#)
- modAlphaCipher, [8](#)
  - convert, [10](#)
  - decrypt, [10](#)
  - encrypt, [12](#)
  - getValidKey, [12](#)
  - getValidText, [13](#)
  - modAlphaCipher, [9](#)
- modAlphaCipher.h, [15](#)