

modAlphaCipher

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
4.2 Класс modAlphaCipher . . . . .	8
4.2.1 Подробное описание . . . . .	9
4.2.2 Конструктор(ы) . . . . .	9
4.2.2.1 modAlphaCipher() . . . . .	9
4.2.3 Методы . . . . .	9
4.2.3.1 convert() [1/2] . . . . .	9
4.2.3.2 convert() [2/2] . . . . .	10
4.2.3.3 decrypt() . . . . .	10
4.2.3.4 encrypt() . . . . .	11
4.2.3.5 getValidKey() . . . . .	11
4.2.3.6 getValidText() . . . . .	12
5 Файлы	13
5.1 Файл modAlphaCipher.h . . . . .	13
5.1.1 Подробное описание . . . . .	14
5.2 modAlphaCipher.h . . . . .	14
Предметный указатель	15



# Глава 1

## Иерархический список классов

### 1.1 Иерархия классов

Иерархия классов.

std::invalid_argument	
cipher_error . . . . .	7
modAlphaCipher . . . . .	8



## Глава 2

# Алфавитный указатель классов

### 2.1 Классы

Классы с их кратким описанием.

<a href="#">cipher_error</a>	
исключений	7
<a href="#">modAlphaCipher</a>	
Класс, осуществляющий 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>	13



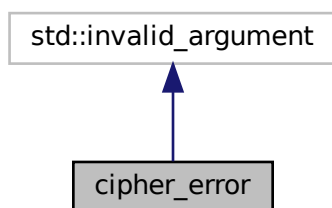
## Глава 4

# Классы

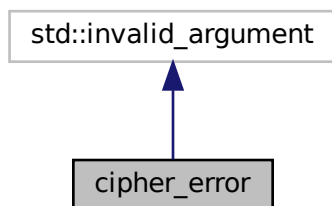
### 4.1 Класс `cipher_error`

исключений.

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

исключений.

Объявления и описания членов класса находятся в файле:

- [modAlphaCipher.h](#)

## 4.2 Класс modAlphaCipher

Класс, осуществляющий шифрование методом "Гронсвельда"

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

### Открытые члены

- modAlphaCipher ()=delete  
Конструктор без параметров.
- [modAlphaCipher](#) (const std::wstring &skey)  
Конструктор для ключа
- std::wstring [encrypt](#) (const std::wstring &open\_text)  
Зашифрование
- std::wstring [decrypt](#) (const std::wstring &cipher\_text)  
Расшифрование

### Закрытые члены

- std::vector< int > [convert](#) (const std::wstring &s)  
Преобразование строк в вектор чисел
- std::wstring [convert](#) (const std::vector< int > &v)  
Преобразование вектора чисел в строку
- std::wstring [getValidKey](#) (const std::wstring &s)  
Проверка ключа
- std::wstring [getValidText](#) (const std::wstring &s)  
проверка текста

### Закрытые данные

- std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"  
Алфавит для текста, использующийся в данной программе
- std::map< char, int > alphaNum  
Ассоциативный массив "номер по символу".
- std::vector< int > key  
Атрибут для ключа

### 4.2.1 Подробное описание

Класс, осуществляющий шифрование методом "Гронсвельда"

### 4.2.2 Конструктор(ы)

#### 4.2.2.1 modAlphaCipher()

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

Конструктор для ключа

Цикл "for" построен по строке-алфавиту

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

Аргументы

строка	текста типа "wstring"
--------	-----------------------

### 4.2.3 Методы

#### 4.2.3.1 convert() [1/2]

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

Преобразование вектора чисел в строку

"wstring" переменная типа с именем "result", в которой формируется строка по индексам алфавита "numAlpha".

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

Возвращает

"result" - строка текста типа "wstring"

#### 4.2.3.2 convert() [2/2]

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

Преобразование строк в вектор чисел

В векторе типа "int" с именем "result" формируются числа

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

Возвращает

std::vector <int>, в котором хранятся индексы букв алфавита

#### 4.2.3.3 decrypt()

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

Расшифрование

Формируется вектор "work" из строки зашифрованного текста с помощью метода [convert\(\)](#). А также зашифрованный текст проверяется на наличие ошибки методом [getValidAlphabetText\(\)](#).

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

Если при зашифровании прибавляется значение ключа, то при расшифровании значения ключа вычитается.

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

Аргументы

wstring	cipher_text - текст расшифрования
---------	-----------------------------------

Исключения

<a href="#">cipher_error</a>	- строка, пришедшая на вход, которая оказывается пустой или в ней есть недопустимые символы
------------------------------	---------------------------------------------------------------------------------------------

Возвращает

строка расшифрованного текста типа "wstring"

## 4.2.3.4 encrypt()

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

## Зашифрование

Формируется вектор "work" из строки текста с помощью метода `convert()`. Происходит проверка текста на наличие ошибки методом `getValidAlphabetText()`.

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

В цикле каждому элементу вектора прибавляется элемент ключа по модулю размера алфавита.

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

## Аргументы

wstring	open_text - текст, для зашифрования
---------	-------------------------------------

## Исключения

<code>cipher_error</code>	- строка, пришедшая на вход, которая оказывается пустой или в ней есть недопустимые символы
---------------------------	---------------------------------------------------------------------------------------------

## Возвращает

строка зашифрованного текста типа "wstring"

## 4.2.3.5 getValidKey()

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

## Проверка ключа

Ключ проверяется на пустоту. Если проверка закончилась успешно, то ключ проверяется на наличие недопустимых символов.

## Исключения

<code>cipher_error</code> , если	ключ является пустым или в нём присутствуют недопустимые символы
----------------------------------	------------------------------------------------------------------

## Возвращает

"result" - строка текста типа "wstring"

#### 4.2.3.6 getValidText()

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

проверка текста

Сначала текст проверяется на пустоту . Если проверка закончилась успешно, то текст проверяется на наличие недопустимых символов.

Исключения

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

Возвращает

строка текста типа "wstring"

Объявления и описания членов класса находятся в файле:

- [modAlphaCipher.h](#)



## Глава 5

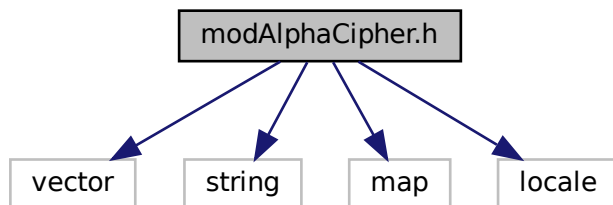
# Файлы

### 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, осуществляющий шифрование методом "Гронсвельда"
- class `cipher_error`  
исключений.

### 5.1.1 Подробное описание

Описание класса `modAlphaCipher`.

Автор

Тюменев Д.С.

Версия

1.0

Дата

20.12.2024

Авторство

ИБСТ ПГУ

## 5.2 `modAlphaCipher.h`

[См. документацию.](#)

```
1
10 #pragma once
11 #include <vector>
12 #include <string>
13 #include <map>
14 #include <locale>
15 using namespace std;
16 class modAlphaCipher
17 {
18 private:
19     std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";
20     std::map <char,int> alphaNum; //ассоциативный массив "номер по символу"
21     std::vector <int> key; //ключ
22     std::vector<int> convert(const std::wstring& s);
23     std::wstring convert(const std::vector<int>& v);
24     std::wstring getValidKey(const std::wstring & s);
25     std::wstring getValidText(const std::wstring & s);
26 public:
27     modAlphaCipher()=delete;
28     modAlphaCipher(const std::wstring& skey);
29     std::wstring encrypt(const std::wstring& open_text);
30     std::wstring decrypt(const std::wstring& cipher_text);
31 };
32
33 class cipher_error: public std::invalid_argument
34 {
35 public:
36     explicit cipher_error (const std::string& what_arg):
37         std::invalid_argument(what_arg) {}
38     explicit cipher_error (const char* what_arg):
39         std::invalid_argument(what_arg) {}
40 };
41
```

# Предметный указатель

- cipher\_error, [7](#)
- convert
  - modAlphaCipher, [9](#)
- decrypt
  - modAlphaCipher, [10](#)
- encrypt
  - modAlphaCipher, [10](#)
- getValidKey
  - modAlphaCipher, [11](#)
- getValidText
  - modAlphaCipher, [11](#)
- modAlphaCipher, [8](#)
  - convert, [9](#)
  - decrypt, [10](#)
  - encrypt, [10](#)
  - getValidKey, [11](#)
  - getValidText, [11](#)
  - modAlphaCipher, [9](#)
- modAlphaCipher.h, [13](#)