

RouteCipher

1.0

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 Класс RouteCipher .....	9
4.2.1 Подробное описание .....	9
4.2.2 Конструктор(ы).....	9
4.2.2.1 RouteCipher().....	9
4.2.3 Методы .....	10
4.2.3.1 decrypt().....	10
4.2.3.2 encrypt() .....	10
4.2.3.3 getKey() .....	11
4.2.3.4 getValidCipherText() .....	11
4.2.3.5 getValidKey().....	12
4.2.3.6 getValidOpenText() .....	12
4.2.3.7 setKey() .....	13
5 Файлы	15
5.1 Файл main.cpp .....	15
5.1.1 Подробное описание .....	16
5.1.2 Функции .....	16
5.1.2.1 isValidKey() .....	16
5.1.2.2 main().....	16
5.2 Файл module.cpp .....	17
5.2.1 Подробное описание .....	17
5.3 Файл module.h .....	18
5.3.1 Подробное описание .....	18
Предметный указатель	21



# 1

## 1.1

Иерархия классов.

std::invalid_argument	
cipher_error .....	<a href="#">7</a>
RouteCipher .....	<a href="#">9</a>



## 2

### 2.1

Классы с их кратким описанием.

<a href="#">cipher_error</a>	Класс исключения для ошибок шифрования .....	<a href="#">7</a>
<a href="#">RouteCipher</a>	Класс для шифрования методом маршрутной перестановки.....	<a href="#">9</a>





### 3.1

Полный список документированных файлов.

<a href="#">main.cpp</a>	Главный модуль программы для шифрования методом маршрутной перестановки	15
<a href="#">module.cpp</a>	Реализация класса <a href="#">RouteCipher</a>	17
<a href="#">module.h</a>	Заголовочный файл класса <a href="#">RouteCipher</a> для шифрования маршрутной перестановкой	18

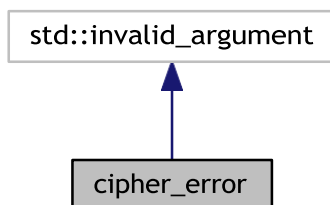


## 4.1 cipher\_error

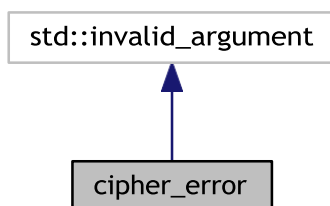
Класс исключения для ошибок шифрования

```
#include <module.h>
```

Граф наследования: cipher\_error:



Граф связей класса cipher\_error:



- [cipher\\_error](#) (const std::string &what\_arg)
- [cipher\\_error](#) (const char \*what\_arg) (C- )

#### 4.1.1

Класс исключения для ошибок шифрования

Наследуется от std::invalid\_argument

#### 4.1.2 ( )

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

```

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

```

Конструктор исключения с передачей строки

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 исключения с передачей строки (C-стиль)

Аргументы

what_arg	- сообщение об ошибке
----------	-----------------------

Объявления и описания членов класса находятся в файле:

- [module.h](#)

## 4.2 RouteCipher

Класс для шифрования методом маршрутной перестановки

#include <module.h>

- [RouteCipher](#) ()=delete
- [RouteCipher](#) (const std::wstring &key)
- void [setKey](#) (const std::wstring &key)
- std::wstring [getKey](#) () const
- std::wstring [encrypt](#) (const std::wstring &text)
- std::wstring [decrypt](#) (const std::wstring &text)
  
- std::wstring [getValidKey](#) (const std::wstring &s)
- std::wstring [getValidOpenText](#) (const std::wstring &s)
- std::wstring [getValidCipherText](#) (const std::wstring &s)
  
- int [columns](#) ( )

### 4.2.1

Класс для шифрования методом маршрутной перестановки

Использует таблицу с заданным числом столбцов. Запись: слева направо, сверху вниз.  
Считывание: сверху вниз, справа налево.

### 4.2.2 ( )

#### 4.2.2.1 RouteCipher()

RouteCipher::RouteCipher (   
 const std::wstring & key )

Конструктор с установкой ключа

Конструктор [RouteCipher](#).

## Аргументы

key	- ключ шифрования (количество столбцов)
-----	---

## Исключения

<a href="#">cipher_error</a>	если ключ невалиден
------------------------------	---------------------

## 4.2.3

## 4.2.3.1 decrypt()

```
std::wstring RouteCipher::decrypt (  
    const std::wstring & text )
```

## Дешифрование текста

## Дешифрование текста методом маршрутной перестановки

## Аргументы

text	- шифртекст
------	-------------

## Возвращает

Расшифрованная строка

## Исключения

<a href="#">cipher_error</a>	если текст пуст или содержит не заглавные буквы
------------------------------	---

## 4.2.3.2 encrypt()

```
std::wstring RouteCipher::encrypt (  
    const std::wstring & text )
```

## Шифрование текста

## Шифрование текста методом маршрутной перестановки

## Аргументы

text	- открытый текст
------	------------------

## Возвращает

Зашифрованная строка

## Исключения

<a href="#">cipher_error</a>	если текст пуст после очистки
------------------------------	-------------------------------

## 4.2.3.3 getKey()

```
std::wstring RouteCipher::getKey ( ) const
```

Получение текущего ключа

## Возвращает

Строковое представление ключа

## 4.2.3.4 getValidCipherText()

```
std::wstring RouteCipher::getValidCipherText (
    const std::wstring & s ) [private]
```

Проверка шифртекста

## Аргументы

s	- шифртекст
---	-------------

## Возвращает

Текст без изменений

## Исключения

<a href="#">cipher_error</a>	если текст пуст или содержит не заглавные буквы
------------------------------	---

#### 4.2.3.5 getValidKey()

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

Проверка и нормализация ключа

Проверка ключа на валидность

Аргументы

s	- исходный ключ (строка с числом)
---	-----------------------------------

Возвращает

Валидная строка ключа

Исключения

<a href="#">cipher_error</a>	если ключ пустой, содержит не цифры, или число $\leq 0$
------------------------------	---

Аргументы

s	- исходный ключ
---	-----------------

Возвращает

Нормализованная строка ключа

Исключения

<a href="#">cipher_error</a>	если ключ невалиден
------------------------------	---------------------

#### 4.2.3.6 getValidOpenText()

```
std::wstring RouteCipher::getValidOpenText (  
    const std::wstring & s ) [private]
```

Проверка и нормализация открытого текста

Проверка и очистка открытого текста

Аргументы

s	- исходный текст
---	------------------



Возвращает

Текст в верхнем регистре без не-букв

Исключения

<a href="#">cipher_error</a>	если после очистки текст пуст
------------------------------	-------------------------------

Аргументы

s	- исходный текст
---	------------------

Возвращает

Текст в верхнем регистре без не-букв

Исключения

<a href="#">cipher_error</a>	если текст пуст после очистки
------------------------------	-------------------------------

#### 4.2.3.7 setKey()

```
void RouteCipher::setKey (  
    const std::wstring & key )
```

Установка нового ключа

Аргументы

key	- новый ключ
-----	--------------

Исключения

<a href="#">cipher_error</a>	если ключ невалиден
------------------------------	---------------------

Объявления и описания членов классов находятся в файлах:

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

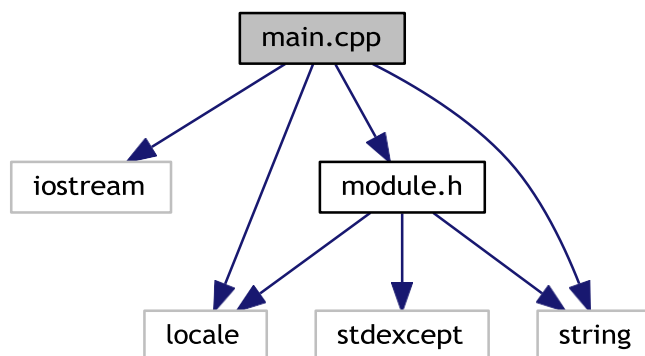


## 5.1 main.cpp

Главный модуль программы для шифрования методом маршрутной перестановки

```
#include <iostream>
#include <locale>
#include <string>
#include "module.h"
```

Граф включаемых заголовочных файлов для main.cpp:



- bool `isValidKey` (const std::wstring &s)
- int `main` ()

### 5.1.1

Главный модуль программы для шифрования методом маршрутной перестановки

([RouteCipher](#))

Автор

Пресняков Александр

Версия

1.0

Дата

2025

Программа предоставляет интерактивный интерфейс для шифрования и дешифрования текста методом табличной маршрутной перестановки.

### 5.1.2

#### 5.1.2.1 isValidKey()

```
bool isValidKey (  
    const std::wstring & s )
```

Проверка корректности ключа

Аргументы

s	- строка с ключом
---	-------------------

Возвращает

true если ключ валиден, false в противном случае

Ключ должен быть положительным целым числом, состоящим только из цифр

#### 5.1.2.2 main()

```
int main ( )
```

Главная функция программы

Возвращает

0 при успешном завершении, 1 при ошибке

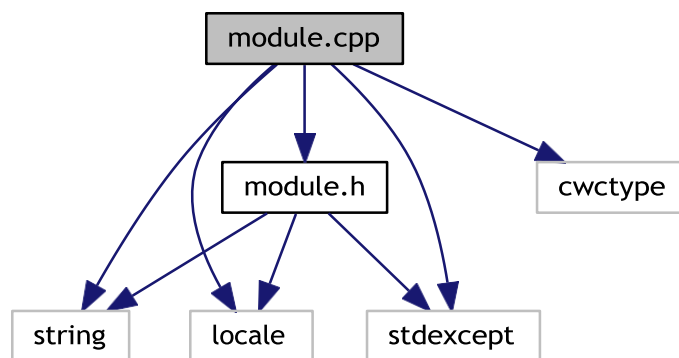
Устанавливает локаль для работы с русским языком, принимает ключ от пользователя, создаёт экземпляр шифра [RouteCipher](#) и предоставляет интерактивное меню для операций шифрования/дешифрования

## 5.2 module.cpp

Реализация класса [RouteCipher](#).

```
#include "module.h"
#include <stdexcept>
#include <locale>
#include <string>
#include <cwctype>
```

Граф включаемых заголовочных файлов для module.cpp:



### 5.2.1

Реализация класса [RouteCipher](#).

Автор

Пресняков Александр

Версия

1.0

Дата

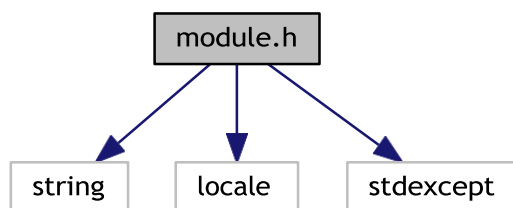
2025

## 5.3 module.h

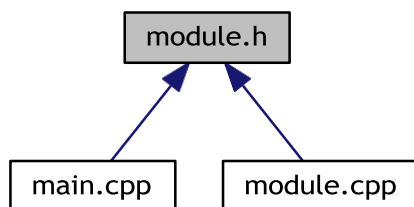
Заголовочный файл класса [RouteCipher](#) для шифрования маршрутной перестановкой

```
#include <string>
#include <locale>
#include <stdexcept>
```

Граф включаемых заголовочных файлов для module.h:



Граф файлов, в которые включается этот файл:



- class [cipher\\_error](#)
- class [RouteCipher](#)

### 5.3.1

Заголовочный файл класса [RouteCipher](#) для шифрования маршрутной перестановкой

Автор

Пресняков Александр

Версия

1.0

Дата

2025

Предупреждения

Реализация для работы с текстами в кодировке UTF-8 (wstring)





- cipher\_error, [7](#)
  - cipher\_error, [8](#)
- decrypt
  - RouteCipher, [10](#)
- encrypt
  - RouteCipher, [10](#)
- getKey
  - RouteCipher, [11](#)
- getValidCipherText
  - RouteCipher, [11](#)
- getValidKey
  - RouteCipher, [11](#)
- getValidOpenText
  - RouteCipher, [12](#)
- isValidKey
  - main.cpp, [16](#)
- main
  - main.cpp, [16](#)
- main.cpp, [15](#)
  - isValidKey, [16](#)
  - main, [16](#)
- module.cpp, [17](#)
- module.h, [18](#)
- RouteCipher, [9](#)
  - decrypt, [10](#)
  - encrypt, [10](#)
  - getKey, [11](#)
  - getValidCipherText, [11](#)
  - getValidKey, [11](#)
  - getValidOpenText, [12](#)
  - RouteCipher, [9](#)
  - setKey, [13](#)
- setKey
  - RouteCipher, [13](#)

CryptoLab -  
1.0

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 Класс modAlphaCipher .....	9
4.2.1 Подробное описание .....	9
4.2.2 Конструктор(ы) .....	10
4.2.2.1 modAlphaCipher() .....	10
4.2.3 Методы .....	10
4.2.3.1 convert() [1/2] .....	10
4.2.3.2 convert() [2/2] .....	11
4.2.3.3 decrypt() .....	11
4.2.3.4 encrypt() .....	12
4.2.3.5 getValidCipherText() .....	13
4.2.3.6 getValidKey() .....	13
4.2.3.7 getValidOpenText() .....	14
5 Файлы	17
5.1 Файл main.cpp .....	17
5.1.1 Подробное описание .....	18
5.1.2 Функции .....	18
5.1.2.1 check() .....	18
5.1.2.2 interactiveMode() .....	18
5.1.2.3 main() .....	19
5.2 Файл modAlphaCipher.cpp .....	19
5.2.1 Подробное описание .....	19
5.3 Файл modAlphaCipher.h .....	20
5.3.1 Подробное описание .....	20
Предметный указатель	23



## 1.1

Иерархия классов.

std::invalid_argument	
cipher_error .....	<a href="#">7</a>
modAlphaCipher .....	<a href="#">9</a>

1

## 2.1

Классы с их кратким описанием.

<a href="#">cipher_error</a>	Класс исключения для ошибок шифрования .....	<a href="#">7</a>
<a href="#">modAlphaCipher</a>	Класс для шифрования методом Гронсфельда .....	<a href="#">9</a>





### 3.1

Полный список документированных файлов.

<a href="#">main.cpp</a>	Главный модуль программы для тестирования шифра Гронсфельда .....	17
<a href="#">modAlphaCipher.cpp</a>	Реализация класса <a href="#">modAlphaCipher</a> .....	19
<a href="#">modAlphaCipher.h</a>	Заголовочный файл класса <a href="#">modAlphaCipher</a> для шифрования методом Гронсфельда .....	20



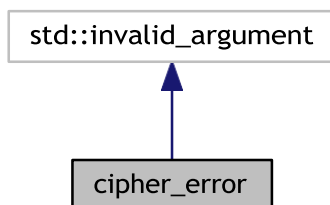


## 4.1 cipher\_error

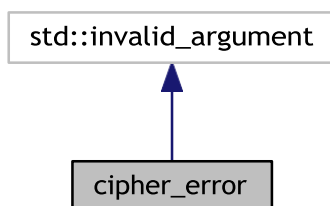
Класс исключения для ошибок шифрования

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

Граф наследования: cipher\_error:



Граф связей класса cipher\_error:



- [cipher\\_error](#) (const std::string &what\_arg)
- [cipher\\_error](#) (const char \*what\_arg) (C- )

#### 4.1.1

Класс исключения для ошибок шифрования

Наследуется от std::invalid\_argument

#### 4.1.2 ( )

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

```

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

```

Конструктор исключения с передачей строки

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 исключения с передачей строки (C-стиль)

Аргументы

what_arg	- сообщение об ошибке
----------	-----------------------

Объявления и описания членов класса находятся в файле:

- [modAlphaCipher.h](#)

## 4.2 modAlphaCipher

Класс для шифрования методом Гронсфельда

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

- `modAlphaCipher ()=delete`
- `modAlphaCipher (const std::wstring &skey)`
- `std::wstring encrypt (const std::wstring &open_text)`
- `std::wstring decrypt (const std::wstring &cipher_text)`
- `std::vector< int > convert (const std::wstring &s)`
- `std::wstring convert (const std::vector< int > &v)`
- `std::wstring getValidKey (const std::wstring &s)`
- `std::wstring getValidOpenText (const std::wstring &s)`
- `std::wstring getValidCipherText (const std::wstring &s)`
- `std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"`
- `std::map< wchar_t, int > alphaNum`  
`"                  ".`
- `std::vector< int > key`

### 4.2.1

Класс для шифрования методом Гронсфельда

Использует сложение символов сообщения с символами ключа по модулю 33 Алфавит: АБВГДЕ-ЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ (33 символа)

## 4.2.2 ( )

### 4.2.2.1 modAlphaCipher()

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

Конструктор с установкой ключа

Конструктор класса

Аргументы

skey	- ключ шифрования
------	-------------------

Исключения

<a href="#">cipher_error</a>	если ключ невалиден
------------------------------	---------------------

Аргументы

skey	- ключ шифрования
------	-------------------

Исключения

<a href="#">cipher_error</a>	если ключ невалиден
------------------------------	---------------------

Инициализирует алфавит и преобразует ключ в числовой вид

## 4.2.3

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

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

Преобразование числового вектора в строку

Аргументы

v	- вектор чисел
---	----------------



Возвращает

Строка, соответствующая вектору

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

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

Преобразование строки в числовой вектор

Аргументы

s	- строка для преобразования
---	-----------------------------

Возвращает

Вектор чисел (номеров символов в алфавите)

Аргументы

s	- строка для преобразования
---	-----------------------------

Возвращает

Вектор номеров символов

#### 4.2.3.3 decrypt()

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

Дешифрование текста

Дешифрование текста методом Гронсфельда

Аргументы

cipher_text	- шифртекст
-------------	-------------

Возвращает

Расшифрованная строка

## Исключения

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

## Аргументы

<code>cipher_text</code>	- шифртекст
--------------------------	-------------

## Возвращает

Расшифрованная строка

## Исключения

<a href="#">cipher_error</a>	если текст пуст или содержит не заглавные буквы
------------------------------	---

Выполняется обратное сложение с ключом по модулю 33

## 4.2.3.4 encrypt()

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

## Шифрование текста

Шифрование текста методом Гронсфельда

## Аргументы

<code>open_text</code>	- открытый текст
------------------------	------------------

## Возвращает

Зашифрованная строка

## Исключения

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

## Аргументы

<code>open_text</code>	- открытый текст
------------------------	------------------

## Возвращает

Зашифрованная строка

Исключения

<code>cipher_error</code>	если текст пуст после очистки
---------------------------	-------------------------------

Текст очищается, преобразуется в числа, складывается с ключом по модулю 33

#### 4.2.3.5 getValidCipherText()

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

Проверка шифртекста

Аргументы

s	- шифртекст
---	-------------

Возвращает

Текст без изменений

Исключения

<code>cipher_error</code>	если текст пуст или содержит не заглавные буквы
---------------------------	---

#### 4.2.3.6 getValidKey()

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

Проверка и нормализация ключа

Аргументы

s	- исходный ключ
---	-----------------

Возвращает

Валидный ключ в верхнем регистре

Исключения

<code>cipher_error</code>	если ключ пустой или содержит не-буквы
---------------------------	--

## Аргументы

s	- исходный ключ
---	-----------------

Возвращает

Ключ в верхнем регистре

Исключения

<a href="#">cipher_error</a>	если ключ пустой или содержит не-буквы
------------------------------	--

4.2.3.7 `getValidOpenText()`

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

Проверка и нормализация открытого текста

Проверка и очистка открытого текста

Аргументы

s	- исходный текст
---	------------------

Возвращает

Текст в верхнем регистре без не-букв

Исключения

<a href="#">cipher_error</a>	если после очистки текст пуст
------------------------------	-------------------------------

Аргументы

s	- исходный текст
---	------------------

Возвращает

Текст в верхнем регистре без не-букв

Исключения

<a href="#">cipher_error</a>	если текст пуст после очистки
------------------------------	-------------------------------

Объявления и описания членов классов находятся в файлах:

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

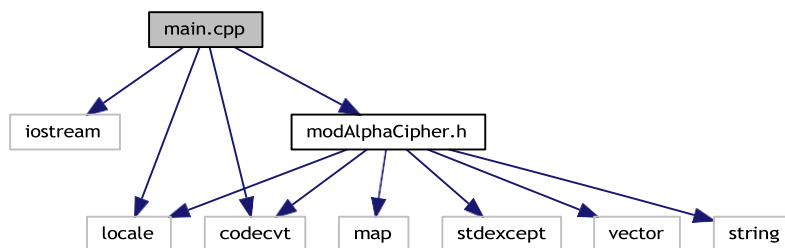


## main.cpp

Главный модуль программы для тестирования шифра Гронсфельда

```
#include <iostream>
#include <locale>
#include <codecvt>
#include "modAlphaCipher.h"
```

Граф включаемых заголовочных файлов для main.cpp:



- void [check](#) (const std::wstring &Text, const std::wstring &key, bool destructCipherText=false)
- void [interactiveMode](#) ()
- int [main](#) ()

Главный модуль программы для тестирования шифра Гронсфельда

Автор

Пресняков Александр

Версия

1.0

Дата

2025

check()

```
void check (
    const std::wstring & Text,
    const std::wstring & key,
    bool destructCipherText = false )
```

Функция для проверки работы шифра

Аргументы

Text	- исходный текст
key	- ключ шифрования
destructCipherText	- флаг для порчи шифртекста (тестирование обработки ошибок)

Шифрует текст, опционально портит его, затем расшифровывает и сравнивает

interactiveMode()

```
void interactiveMode ( )
```

Интерактивный режим работы программы

Позволяет пользователю вводить ключ, шифровать/дешифровать тексты и запускать автоматическое тестирование



main()

int main ( )

Главная функция программы

Возвращает

0 при успешном завершении

Устанавливает локаль и запускает интерактивный режим

## modAlphaCipher.cpp

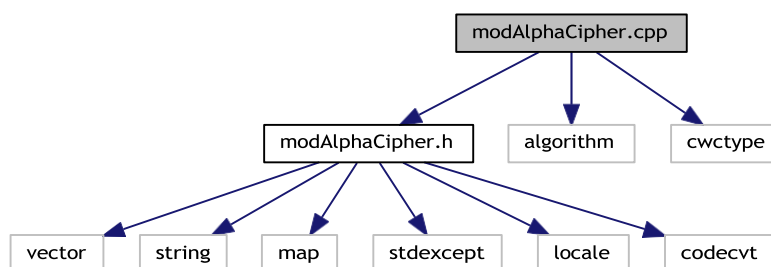
Реализация класса [modAlphaCipher](#).

```
#include "modAlphaCipher.h"
```

```
#include <algorithm>
```

```
#include <cwctype>
```

Граф включаемых заголовочных файлов для modAlphaCipher.cpp:



Реализация класса [modAlphaCipher](#).

Автор

Пресняков Александр

Версия

1.0

Дата

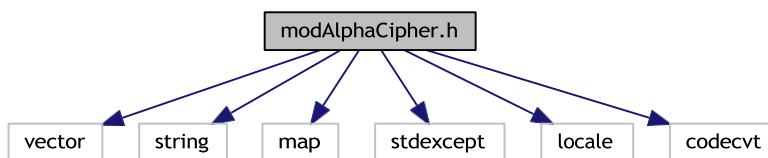
2025

## modAlphaCipher.h

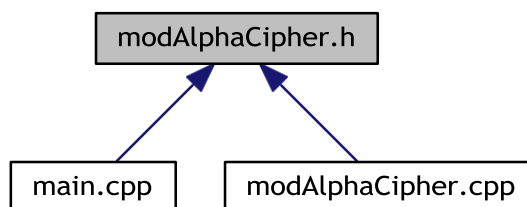
Заголовочный файл класса `modAlphaCipher` для шифрования методом Гронсфельда

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

Граф включаемых заголовочных файлов для `modAlphaCipher.h`:



Граф файлов, в которые включается этот файл:



- class `cipher_error`
- class `modAlphaCipher`

Заголовочный файл класса `modAlphaCipher` для шифрования методом Гронсфельда

Автор

Пресняков Александр

Версия

1.0

Дата

2025

Предупреждения

Реализация для русского алфавита (33 буквы)



- check
  - main.cpp, [18](#)
- cipher\_error, [7](#)
  - cipher\_error, [8](#)
- convert
  - modAlphaCipher, [10](#), [11](#)
- decrypt
  - modAlphaCipher, [11](#)
- encrypt
  - modAlphaCipher, [12](#)
- getValidCipherText
  - modAlphaCipher, [13](#)
- getValidKey
  - modAlphaCipher, [13](#)
- getValidOpenText
  - modAlphaCipher, [14](#)
- interactiveMode
  - main.cpp, [18](#)
- main
  - main.cpp, [18](#)
- main.cpp, [17](#)
  - check, [18](#)
  - interactiveMode, [18](#)
  - main, [18](#)
- modAlphaCipher, [9](#)
  - convert, [10](#), [11](#)
  - decrypt, [11](#)
  - encrypt, [12](#)
  - getValidCipherText, [13](#)
  - getValidKey, [13](#)
  - getValidOpenText, [14](#)
  - modAlphaCipher, [10](#)
- modAlphaCipher.cpp, [19](#)
- modAlphaCipher.h, [20](#)

Ссылка на репозиторий : [https://github.com/24pt208/lab\\_4](https://github.com/24pt208/lab_4)