

Документация

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() [1/4]	8
4.1.2.2 cipher_error() [2/4]	8
4.1.2.3 cipher_error() [3/4]	8
4.1.2.4 cipher_error() [4/4]	8
4.2 Класс code	9
4.2.1 Подробное описание	9
4.2.2 Конструктор(ы)	9
4.2.2.1 code() [1/2]	9
4.2.2.2 code() [2/2]	10
4.2.3 Методы	11
4.2.3.1 encryption()	11
4.2.3.2 getValidCipherText()	12
4.2.3.3 getValidKey()	13
4.2.3.4 getValidOpenText()	13
4.2.3.5 transcript()	14
4.2.4 Данные класса	15
4.2.4.1 key	15
4.3 Структура KeyThree_fixture	15
4.3.1 Подробное описание	16
4.3.2 Конструктор(ы)	16
4.3.2.1 KeyThree_fixture()	16
4.3.2.2 ~KeyThree_fixture()	16
4.3.3 Данные класса	16
4.3.3.1 t	16
4.4 Класс modAlphaCipher	16
4.4.1 Подробное описание	17
4.4.2 Конструктор(ы)	17
4.4.2.1 modAlphaCipher() [1/2]	18
4.4.2.2 modAlphaCipher() [2/2]	18
4.4.3 Методы	18
4.4.3.1 convert() [1/2]	18

4.4.3.2 convert() [2/2]	19
4.4.3.3 decrypt()	19
4.4.3.4 encrypt()	20
4.4.3.5 getValidCipherText()	20
4.4.3.6 getValidKey()	21
4.4.3.7 getValidOpenText()	21
4.4.4 Данные класса	22
4.4.4.1 alphaNum	22
4.4.4.2 key	22
4.4.4.3 numAlpha	22
4.5 Структура SimpleFixture	22
4.5.1 Подробное описание	23
4.5.2 Конструктор(ы)	23
4.5.2.1 SimpleFixture()	23
4.5.2.2 ~SimpleFixture()	23
4.5.3 Данные класса	23
4.5.3.1 p	23
5 Файлы	25
5.1 Файл 1/main.cpp	25
5.1.1 Подробное описание	26
5.1.2 Функции	26
5.1.2.1 main()	26
5.1.2.2 SUITE() [1/3]	26
5.1.2.3 SUITE() [2/3]	27
5.1.2.4 SUITE() [3/3]	27
5.2 Файл 2/main.cpp	27
5.2.1 Подробное описание	28
5.2.2 Функции	28
5.2.2.1 main()	28
5.2.2.2 SUITE() [1/3]	28
5.2.2.3 SUITE() [2/3]	29
5.2.2.4 SUITE() [3/3]	29
5.3 Файл 1/modAlphaCIpsher.cpp	29
5.3.1 Переменные	29
5.3.1.1 codec	30
5.4 Файл 1/modAlphaCipher.h	30
5.4.1 Подробное описание	31
5.5 modAlphaCipher.h	31
5.6 Файл 2/route.cpp	32
5.6.1 Подробное описание	32
5.7 Файл 2/route.h	32
5.7.1 Подробное описание	33

5.8 route.h	34
Предметный указатель	35

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

code	9
std::invalid_argument	
cipher_error	7
cipher_error	7
KeyThree_fixture	15
modAlphaCipher	16
SimpleFixture	22

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

cipher_error		
	Класс исключения для ошибок шифрования	7
code		
	Класс для шифрования и расшифрования методом табличной маршрутной перестановки	9
KeyThree_fixture		
	Фикстура для тестов с ключом 3	15
modAlphaCipher		
	Класс для шифрования и расшифрования текста методом Гронсфельда (русский алфавит)	16
SimpleFixture		
	Фикстура для тестов шифрования/расшифрования	22

Глава 3

Список файлов

3.1 Файлы

Полный список файлов.

1/main.cpp	Модульные тесты для класса modAlphaCipher (UnitTest++)	25
1/modAlphaCipher.cpp		29
1/modAlphaCipher.h	Заголовочный файл для модуля шифрования методом Гронсфельда (русская версия)	30
2/main.cpp	Модульные тесты для класса code (UnitTest++)	27
2/route.cpp	Реализация класса code для шифрования методом табличной маршрутной перестановки	32
2/route.h	Заголовочный файл для модуля шифрования методом табличной маршрутной перестановки	32

Глава 4

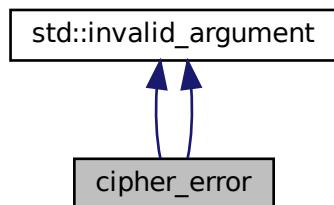
Классы

4.1 Класс cipher_error

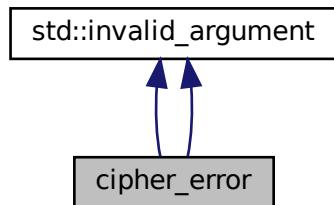
Класс исключения для ошибок шифрования

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

Граф наследования:cipher_error:



Граф связей класса cipher_error:



Открытые члены

- [cipher_error \(const std::string &what_arg\)](#)
- [cipher_error \(const char *what_arg\)](#)
- [cipher_error \(const string &what_arg\)](#)
- [cipher_error \(const char *what_arg\)](#)

4.1.1 Подробное описание

Класс исключения для ошибок шифрования

Производный от `std::invalid_argument`, используется для обработки ошибок в модуле шифрования

4.1.2 Конструктор(ы)

4.1.2.1 cipher_error() [1/4]

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

4.1.2.2 cipher_error() [2/4]

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

4.1.2.3 cipher_error() [3/4]

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

4.1.2.4 cipher_error() [4/4]

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

Объявления и описания членов классов находятся в файлах:

- 1/[modAlphaCipher.h](#)
- 2/[route.h](#)

4.2 Класс code

Класс для шифрования и расшифрования методом табличной маршрутной перестановки

```
#include <route.h>
```

Открытые члены

- `code ()=delete`
Запрет конструктора без параметров
- `code (int skey, string text)`
Конструктор с ключом и текстом
- `string encryption (const string &text)`
Шифрует открытый текст методом маршрутной перестановки
- `string transcript (const string &text, const string &open_text)`
Расшифровывает зашифрованный текст

Закрытые члены

- `int getValidKey (int key, const string &Text)`
Проверяет и возвращает валидный ключ
- `string getValidOpenText (const string &s)`
Проверяет и нормализует открытый текст
- `string getValidCipherText (const string &s, const string &open_text)`
Проверяет зашифрованный текст на соответствие длине

Закрытые данные

- `int key`
Количество столбцов таблицы (ключ шифрования)

4.2.1 Подробное описание

Класс для шифрования и расшифрования методом табличной маршрутной перестановки

Использует таблицу с заданным количеством столбцов (ключ). Запись: по строкам слева направо, сверху вниз. Чтение: по столбцам сверху вниз, справа налево.

4.2.2 Конструктор(ы)

4.2.2.1 code() [1/2]

```
code::code ( ) [delete]
```

Запрет конструктора без параметров

4.2.2.2 code() [2/2]

```
code::code (  
    int skey,  
    string text )
```

Конструктор с ключом и текстом

Конструктор класса code.

Аргументы

skey	Ключ шифрования (количество столбцов)
text	Открытый текст для проверки длины ключа

Исключения

cipher_error	Если ключ некорректного размера
--------------	---------------------------------

Аргументы

skey	Ключ шифрования
text	Открытый текст

Исключения

cipher_error	Если ключ некорректного размера
--------------	---------------------------------

4.2.3 Методы

4.2.3.1 encryption()

```
string code::encryption (  
    const string & text )
```

Шифрует открытый текст методом маршрутной перестановки

Шифрование текста методом табличной маршрутной перестановки

Аргументы

text	Текст для шифрования
------	----------------------

Возвращает

Зашифрованная строка

Исключения

cipher_error	Если текст пустой или содержит некорректные символы
--------------	---

Аргументы

text	Открытый текст
------	----------------

Возвращает

Зашифрованная строка

4.2.3.2 getValidCipherText()

```
string code::getValidCipherText (
    const string & s,
    const string & open_text ) [inline], [private]
```

Проверяет зашифрованный текст на соответствие длине

Аргументы

s	Зашифрованный текст
open_text	Исходный открытый текст

Возвращает

Зашифрованный текст без изменений

Исключения

cipher_error	Если длины текстов не совпадают
--------------	---------------------------------

Аргументы

s	Зашифрованный текст
open_text	Открытый текст

Возвращает

Зашифрованный текст

Исключения

cipher_error	Если длины не совпадают
--------------	-------------------------

4.2.3.3 getValidKey()

```
int code::getValidKey (
    int key,
    const string & Text ) [inline], [private]
```

Проверяет и возвращает валидный ключ

Проверяет валидность ключа

Аргументы

key	Предлагаемый ключ
Text	Открытый текст для проверки длины

Возвращает

Валидный ключ

Исключения

cipher_error	Если ключ некорректного размера
--------------	---------------------------------

Аргументы

key	Предлагаемый ключ
Text	Открытый текст

Возвращает

Валидный ключ

Исключения

cipher_error	Если ключ меньше 2 или больше длины текста
--------------	--

4.2.3.4 getValidOpenText()

```
string code::getValidOpenText (
    const string & s ) [inline], [private]
```

Проверяет и нормализует открытый текст

Аргументы

s	Исходный открытый текст
---	-------------------------

Возвращает

Текст без пробелов и с проверкой символов

Исключения

cipher_error	Если текст пустой или содержит некорректные символы
------------------------------	---

Аргументы

s	Открытый текст
---	----------------

Возвращает

Текст без пробелов

Исключения

cipher_error	Если текст пустой или содержит некорректные символы
------------------------------	---

4.2.3.5 transcript()

```
string code::transcript (
    const string & text,
    const string & open_text )
```

Расшифровывает зашифрованный текст

Расшифрование текста

Аргументы

text	Зашифрованный текст
open_text	Исходный открытый текст (для проверки длины)

Возвращает

Расшифрованная строка

Исключения

cipher_error	Если тексты пустые, содержат некорректные символы или разной длины
------------------------------	--

Аргументы

text	Зашифрованный текст
open_text	Исходный открытый текст

Возвращает

Расшифрованная строка

4.2.4 Данные класса

4.2.4.1 key

```
int code::key [private]
```

Количество столбцов таблицы (ключ шифрования)

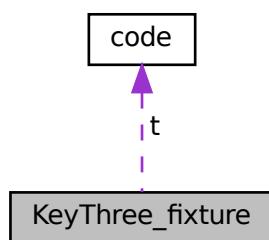
Объявления и описания членов классов находятся в файлах:

- 2/[route.h](#)
- 2/[route.cpp](#)

4.3 Структура KeyThree_fixture

Фикстура для тестов с ключом 3.

Граф связей класса KeyThree_fixture:



Открытые члены

- [KeyThree_fixture \(\)](#)
- [~KeyThree_fixture \(\)](#)

Открытые атрибуты

- `code * t`

4.3.1 Подробное описание

Фикстура для тестов с ключом 3.

4.3.2 Конструктор(ы)

4.3.2.1 KeyThree_fixture()

`KeyThree_fixture::KeyThree_fixture () [inline]`

4.3.2.2 ~KeyThree_fixture()

`KeyThree_fixture::~KeyThree_fixture () [inline]`

4.3.3 Данные класса

4.3.3.1 t

`code* KeyThree_fixture::t`

Объявления и описания членов структуры находятся в файле:

- 2/[main.cpp](#)

4.4 Класс modAlphaCipher

Класс для шифрования и расшифрования текста методом Гронсфельда (русский алфавит)

`#include <modAlphaCipher.h>`

Открытые члены

- `modAlphaCipher ()=delete`
Запрет конструктора без параметров
- `modAlphaCipher (const std::string &skey)`
Конструктор с ключом
- `std::string encrypt (const std::string &open_text)`
Шифрует открытый текст
- `std::string decrypt (const std::string &cipher_text)`
Расшифровывает зашифрованный текст

Закрытые члены

- `std::vector< int > convert (const std::string &s)`
Преобразует строку в вектор числовых кодов
- `std::string convert (const std::vector< int > &v)`
Преобразует вектор числовых кодов в строку
- `std::string getValidKey (const std::string &s)`
Проверяет и нормализует ключ
- `std::string getValidOpenText (const std::string &s)`
Проверяет и нормализует открытый текст
- `std::string getValidCipherText (const std::string &s)`
Проверяет зашифрованный текст

Закрытые данные

- `std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОРСТУФХ҆ЧШЩЬЫЭЮЯ"`
Алфавит по порядку
- `std::map< wchar_t, int > alphaNum`
Ассоциативный массив "символ -> номер".
- `std::vector< int > key`
Ключ в числовом виде

4.4.1 Подробное описание

Класс для шифрования и расшифрования текста методом Гронсфельда (русский алфавит)

Использует русский алфавит из 33 букв (А-Я, Ё). Ключ и текст должны содержать только русские буквы (регистр не важен).

4.4.2 Конструктор(ы)

4.4.2.1 modAlphaCipher() [1/2]

modAlphaCipher::modAlphaCipher () [delete]

Запрет конструктора без параметров

4.4.2.2 modAlphaCipher() [2/2]

modAlphaCipher::modAlphaCipher (const std::string & skey)

Конструктор с ключом

Конструктор класса modAlphaCipher.

Аргументы

skey	Ключ в виде строки
------	--------------------

Исключения

cipher_error	Если ключ пустой, содержит не-буквы или является слабым (одинаковые символы)
--------------	--

Аргументы

skey	Ключ шифрования
------	-----------------

Исключения

cipher_error	При недопустимом ключе
--------------	------------------------

4.4.3 Методы

4.4.3.1 convert() [1/2]

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

Преобразует строку в вектор числовых кодов

Аргументы

s	Входная строка
---	----------------

Возвращает

Вектор номеров символов в алфавите

4.4.3.2 convert() [2/2]

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

Преобразует вектор числовых кодов в строку

Аргументы

v	Вектор номеров символов
---	-------------------------

Возвращает

Строка, соответствующая вектору

4.4.3.3 decrypt()

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

Расшифровывает зашифрованный текст

Расшифрование текста

Аргументы

cipher_text	Зашифрованный текст
-------------	---------------------

Возвращает

Расшифрованная строка (в верхнем регистре)

Исключения

cipher_error	Если текст пустой или содержит недопустимые символы
--------------	---

Аргументы

cipher_text	Зашифрованный текст
-------------	---------------------

Возвращает

Расшифрованная строка

4.4.3.4 encrypt()

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

Шифрует открытый текст

Шифрование текста

Аргументы

open_text	Текст для шифрования
-----------	----------------------

Возвращает

Зашифрованная строка (в верхнем регистре)

Исключения

cipher_error	Если текст пустой или не содержит букв
--------------	--

Аргументы

open_text	Открытый текст
-----------	----------------

Возвращает

Зашифрованная строка

4.4.3.5 getValidCipherText()

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

Проверяет зашифрованный текст

Аргументы

s	Зашифрованный текст
---	---------------------

Возвращает

Текст без изменений

Исключения

cipher_error	Если текст пустой или содержит недопустимые символы
--------------	---

4.4.3.6 getValidKey()

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

Проверяет и нормализует ключ

Аргументы

s	Исходный ключ
---	---------------

Возвращает

Ключ в верхнем регистре без не-букв

Исключения

cipher_error	Если ключ пустой или содержит не-буквы
--------------	--

4.4.3.7 getValidOpenText()

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

Проверяет и нормализует открытый текст

Аргументы

s	Исходный открытый текст
---	-------------------------

Возвращает

Текст в верхнем регистре без не-букв

Исключения

cipher_error	Если текст пустой после удаления не-букв
------------------------------	--

4.4.4 Данные класса

4.4.4.1 alphaNum

```
std::map<wchar_t,int> modAlphaCipher::alphaNum [private]
```

Ассоциативный массив "символ -> номер".

4.4.4.2 key

```
std::vector<int> modAlphaCipher::key [private]
```

Ключ в числовом виде

4.4.4.3 numAlpha

```
std::wstring modAlphaCipher::numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОРСТУФХЦЧШЩЪЫЬЭЮЯ" [private]
```

Алфавит по порядку

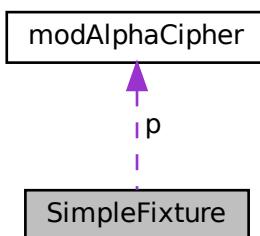
Объявления и описания членов классов находятся в файлах:

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

4.5 Структура SimpleFixture

Фикстура для тестов шифрования/расшифрования

Граф связей класса SimpleFixture:



Открытые члены

- `SimpleFixture ()`
- `~SimpleFixture ()`

Открытые атрибуты

- `modAlphaCipher * p`

4.5.1 Подробное описание

Фикстура для тестов шифрования/расшифрования

4.5.2 Конструктор(ы)

4.5.2.1 `SimpleFixture()`

`SimpleFixture::SimpleFixture ()` [inline]

4.5.2.2 `~SimpleFixture()`

`SimpleFixture::~SimpleFixture ()` [inline]

4.5.3 Данные класса

4.5.3.1 `p`

`modAlphaCipher* SimpleFixture::p`

Объявления и описания членов структуры находятся в файле:

- 1/[main.cpp](#)

Глава 5

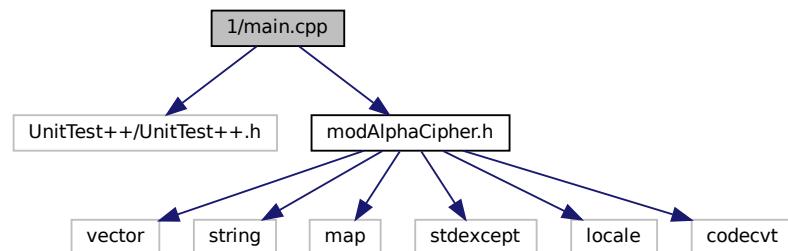
Файлы

5.1 Файл 1/main.cpp

Модульные тесты для класса modAlphaCipher (UnitTest++)

```
#include <UnitTest++/UnitTest++.h>
#include "modAlphaCipher.h"
```

Граф включаемых заголовочных файлов для main.cpp:



Классы

- struct SimpleFixture
Фикстура для тестов шифрования/расшифрования

Функции

- SUITE (KeyTest)
Тесты для конструктора и ключа
- SUITE (EncryptTest)
Тесты шифрования
- SUITE (DecryptTest)
Тесты расшифрования
- int main (int argc, char **argv)
Главная функция запуска тестов

5.1.1 Подробное описание

Модульные тесты для класса `modAlphaCipher` (UnitTest++)

Автор

Никита Седнёв

Версия

1.0

Дата

2025-11-27

Тестирование конструктора, шифрования и расшифрования

5.1.2 Функции

5.1.2.1 main()

```
int main (
    int argc,
    char ** argv )
```

Главная функция запуска тестов

Аргументы

argc	Количество аргументов
argv	Аргументы командной строки

Возвращает

Код завершения тестирования

5.1.2.2 SUITE() [1/3]

```
SUITE (
    DecryptTest )
```

Тесты расшифрования

5.1.2.3 SUITE() [2/3]

```
SUITE (
    EncryptTest )
```

Тесты шифрования

5.1.2.4 SUITE() [3/3]

```
SUITE (
    KeyTest )
```

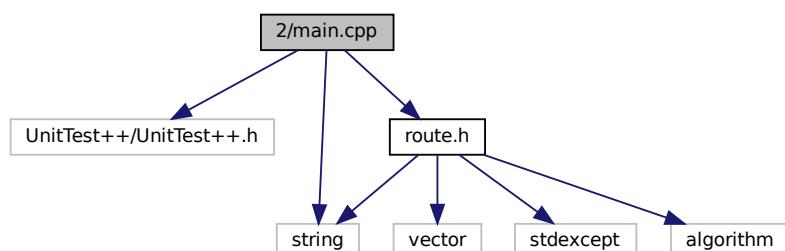
Тесты для конструктора и ключа

5.2 Файл 2/main.cpp

Модульные тесты для класса code (UnitTest++)

```
#include <UnitTest++/UnitTest++.h>
#include "route.h"
#include <string>
```

Граф включаемых заголовочных файлов для main.cpp:



Классы

- struct **KeyThree_fixture**

Фикстура для тестов с ключом 3.

Функции

- **SUITE (KeyTest)**
Тесты для конструктора и ключа
- **SUITE (EncryptTest)**
Тесты шифрования
- **SUITE (DecryptTest)**
Тесты расшифрования
- int **main** (int argc, char **argv)
Главная функция запуска тестов

5.2.1 Подробное описание

Модульные тесты для класса code (UnitTest++)

Автор

Ваше Имя

Версия

1.0

Дата

2025-12-09

Тестирование конструктора, шифрования и расшифрования методом табличной маршрутной перестановки

5.2.2 Функции

5.2.2.1 main()

```
int main (
    int argc,
    char ** argv )
```

Главная функция запуска тестов

Аргументы

argc	Количество аргументов
argv	Аргументы командной строки

Возвращает

Код завершения тестирования

5.2.2.2 SUITE() [1/3]

```
SUITE (
    DecryptTest )
```

Тесты расшифрования

5.2.2.3 SUITE() [2/3]

```
SUITE (
    EncryptTest )
```

Тесты шифрования

5.2.2.4 SUITE() [3/3]

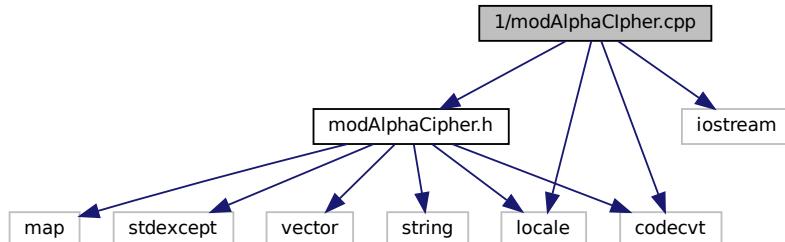
```
SUITE (
    KeyTest )
```

Тесты для конструктора и ключа

5.3 Файл 1/modAlphaCIpher.cpp

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

Граф включаемых заголовочных файлов для modAlphaCIpher.cpp:



Переменные

- std::wstring_convert< std::codecvt_utf8< wchar_t >, wchar_t > **codec**
Конвертер UTF-8 <-> wstring.

5.3.1 Переменные

5.3.1.1 codec

```
std::wstring_convert<std::codecvt_utf8<wchar_t>, wchar_t> codec
```

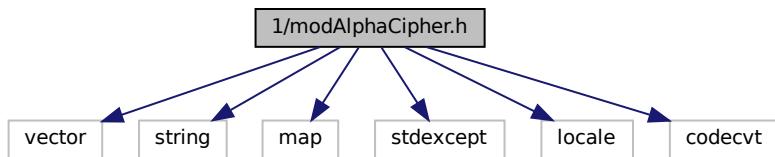
Конвертер UTF-8 <-> wstring.

5.4 Файл 1/modAlphaCipher.h

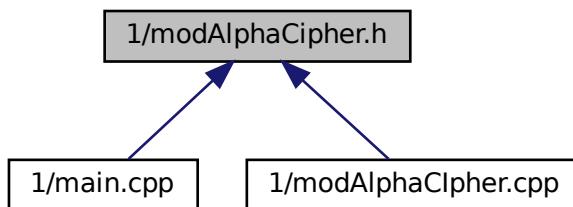
Заголовочный файл для модуля шифрования методом Гронсфельда (русская версия)

```
#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:



Классы

- class **cipher_error**

Класс исключения для ошибок шифрования

- class **modAlphaCipher**

Класс для шифрования и расшифрования текста методом Гронсфельда (русский алфавит)

5.4.1 Подробное описание

Заголовочный файл для модуля шифрования методом Гронсфельда (русская версия)

Автор

Никита Седнёв

Версия

1.0

Дата

2025-11-27

Предупреждения

Реализация только для русского языка (алфавит: А-Я, Ё)

5.5 modAlphaCipher.h

[См. документацию.](#)

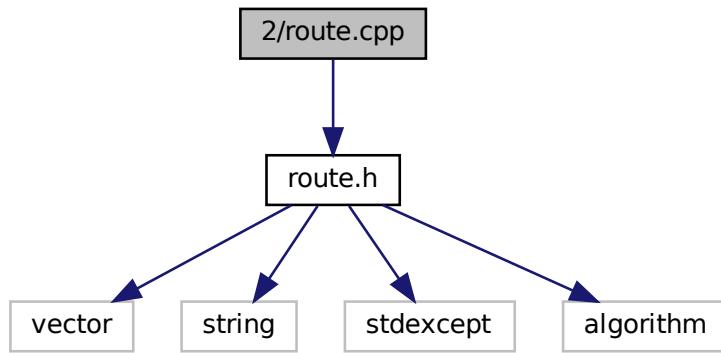
```
1
10 #pragma once
11 #include <vector>
12 #include <string>
13 #include <map>
14 #include <stdexcept>
15 #include <locale>
16 #include <codecvt>
17
23 class cipher_error: public std::invalid_argument {
24 public:
25     explicit cipher_error(const std::string& what_arg):
26         std::invalid_argument(what_arg) {}
27     explicit cipher_error(const char* what_arg):
28         std::invalid_argument(what_arg) {}
29 };
30
37 class modAlphaCipher {
38 private:
39     std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";
40     std::map<wchar_t,int> alphaNum;
41     std::vector<int> key;
42
43     std::vector<int> convert(const std::string& s);
44     std::string convert(const std::vector<int>& v);
45     std::string getValidKey(const std::string & s);
46     std::string getValidOpenText(const std::string & s);
47     std::string getValidCipherText(const std::string & s);
48
49 public:
50     modAlphaCipher() = delete;
51
57     modAlphaCipher(const std::string& skey);
58
65     std::string encrypt(const std::string& open_text);
66
73     std::string decrypt(const std::string& cipher_text);
74 };
```

5.6 Файл 2/route.cpp

Реализация класса code для шифрования методом табличной маршрутной перестановки

```
#include "route.h"
```

Граф включаемых заголовочных файлов для route.cpp:



5.6.1 Подробное описание

Реализация класса code для шифрования методом табличной маршрутной перестановки

Автор

Ваше Имя

Версия

1.0

Дата

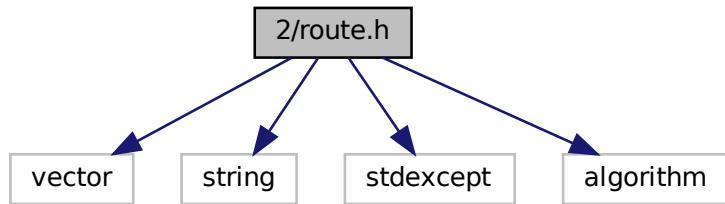
2025-12-09

5.7 Файл 2/route.h

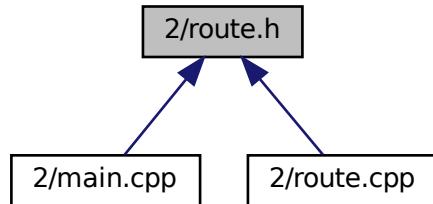
Заголовочный файл для модуля шифрования методом табличной маршрутной перестановки

```
#include <vector>
#include <string>
#include <stdexcept>
```

```
#include <algorithm>
Граф включаемых заголовочных файлов для route.h:
```



Граф файлов, в которые включается этот файл:



Классы

- class `cipher_error`
Класс исключения для ошибок шифрования
- class `code`
Класс для шифрования и расшифрования методом табличной маршрутной перестановки

5.7.1 Подробное описание

Заголовочный файл для модуля шифрования методом табличной маршрутной перестановки

Автор

Ваше Имя

Версия

1.0

Дата

2025-12-09

Предупреждения

Реализация только для английского языка

5.8 route.h

[См. документацию.](#)

```
1
10 #pragma once
11 #include <vector>
12 #include <string>
13 #include <stdexcept>
14 #include <algorithm>
15 using namespace std;
16
22 class cipher_error: public invalid_argument {
23 public:
24     explicit cipher_error(const string& what_arg):
25         invalid_argument(what_arg) {}
26     explicit cipher_error(const char* what_arg):
27         invalid_argument(what_arg) {}
28 };
29
37 class code {
38 private:
39     int key;
40
48     inline int getValidKey(int key, const string& Text);
49
56     inline string getValidOpenText(const string& s);
57
65     inline string getValidCipherText(const string& s, const string& open_text);
66
67 public:
68     code() = delete;
69
76     code(int skey, string text);
77
84     string encryption(const string& text);
85
93     string transcript(const string& text, const string& open_text);
94 };
```

Предметный указатель

~KeyThree_fixture
 KeyThree_fixture, 16

~SimpleFixture
 SimpleFixture, 23

1/main.cpp, 25

1/modAlphaCIpsher.cpp, 29

1/modAlphaCipher.h, 30, 31

2/main.cpp, 27

2/route.cpp, 32

2/route.h, 32, 34

alphaNum
 modAlphaCipher, 22

cipher_error, 7
 cipher_error, 8

code, 9
 code, 9

 encryption, 11

 getValidCipherText, 12

 getValidKey, 12

 getValidOpenText, 13

 key, 15

 transcript, 14

codec
 modAlphaCIpsher.cpp, 29

convert
 modAlphaCipher, 18, 19

decrypt
 modAlphaCipher, 19

encrypt
 modAlphaCipher, 20

encryption
 code, 11

getValidCipherText
 code, 12
 modAlphaCipher, 20

getValidKey
 code, 12
 modAlphaCipher, 21

getValidOpenText
 code, 13
 modAlphaCipher, 21

key
 code, 15
 modAlphaCipher, 22

KeyThree_fixture, 15
 ~KeyThree_fixture, 16

 KeyThree_fixture, 16

t, 16

main
 main.cpp, 26, 28

main.cpp
 main, 26, 28

 SUITE, 26–29

modAlphaCipher, 16
 alphaNum, 22

 convert, 18, 19

 decrypt, 19

 encrypt, 20

 getValidCipherText, 20

 getValidKey, 21

 getValidOpenText, 21

 key, 22

 modAlphaCipher, 17, 18

 numAlpha, 22

modAlphaCIpsher.cpp
 codec, 29

numAlpha
 modAlphaCipher, 22

p
 SimpleFixture, 23

SimpleFixture, 22
 ~SimpleFixture, 23

 p, 23

 SimpleFixture, 23

SUITE
 main.cpp, 26–29

t
 KeyThree_fixture, 16

transcript
 code, 14