Шифрование: Гронсфельд и табличная перестановка

Создано системой Doxygen 1.9.4

1 lab-2 1

2 lab-4 3

3 Иерархический список классов 5
3.1 Иерархия классов . . . 5

4 Алфавитный указатель классов 7
4.1 Классы . . . 7

5 Список файлов 9
5.1 Файлы . . . 9

6 Классы 11
6.1 Класс modAlphaCipher . . . 11
6.1.1 Подробное описание . . . 12
6.1.2 Конструктор(ы) . . . 12
6.1.2.1 modAlphaCipher() [1/2] . . . 12
6.1.2.2 modAlphaCipher() [2/2] . . . 12
6.1.3 Методы . . . 13
6.1.3.1 convert() [1/2] . . . 13
6.1.3.2 convert() [2/2] . . . 13
6.1.3.3 decrypt() . . . 14
6.1.3.4 encrypt() . . . 15
6.1.3.5 getValidCipherText() . . . 16
6.1.3.6 getValidKey() . . . 16
6.1.3.7 getValidOpenText() . . . 17
6.1.4 Данные класса . . . 18
6.1.4.1 alphaNum . . . 18
6.1.4.2 key . . . 18
6.1.4.3 numAlpha . . . 18
6.2 Класс route_cipher_error . . . 19
6.2.1 Подробное описание . . . 19
6.2.2 Конструктор(ы) . . . 20
6.2.2.1 route_cipher_error() [1/2] . . . 20
6.2.2.2 route_cipher_error() [2/2] . . . 20
6.3 Класс routeCipher . . . 20
6.3.1 Подробное описание . . . 21
6.3.2 Конструктор(ы) . . . 22
6.3.2.1 routeCipher() [1/4] . . . 22
6.3.2.2 routeCipher() [2/4] . . . 22
6.3.2.3 routeCipher() [3/4] . . . 23
6.3.2.4 routeCipher() [4/4] . . . 23
6.3.3 Методы . . . 23
6.3.3.1 createTable() [1/2] . . . 23

6.3.3.2 createTable() [2/2] . . . 23
6.3.3.3 decrypt() [1/2] . . . 24
6.3.3.4 decrypt() [2/2] . . . 24
6.3.3.5 encrypt() [1/2] . . . 24
6.3.3.6 encrypt() [2/2] . . . 25
6.3.3.7 getValidCipherText() [1/2] . . . 26
6.3.3.8 getValidCipherText() [2/2] . . . 26
6.3.3.9 getValidOpenText() [1/2] . . . 26
6.3.3.10 getValidOpenText() [2/2] . . . 27
6.3.3.11 prepareText() [1/2] . . . 27
6.3.3.12 prepareText() [2/2] . . . 27
6.3.3.13 readDecrypted() [1/2] . . . 28
6.3.3.14 readDecrypted() [2/2] . . . 28
6.3.3.15 readEncrypted() [1/2] . . . 28
6.3.3.16 readEncrypted() [2/2] . . . 28
6.3.3.17 validateColumns() [1/2] . . . 29
6.3.3.18 validateColumns() [2/2] . . . 29
6.3.4 Данные класса . . . 29
6.3.4.1 columns . . . 29

7 Файлы 31
7.1 Файл lab-3/task1/modAlphaCipher.cpp . . . 31
7.1.1 Подробное описание . . . 31
7.2 Файл lab-3/task1/modAlphaCipher.h . . . 32
7.2.1 Переменные . . . 33
7.2.1.1 смотри . . . 33
7.2.1.2 columns . . . 33
7.2.1.3 cpp . . . 33
7.2.1.4 h . . . 33
7.2.1.5 task1 . . . 33
7.3 modAlphaCipher.h . . . 34
7.4 Файл lab-3/task2/routeCipher.cpp . . . 38
7.4.1 Подробное описание . . . 39
7.5 Файл lab-3/task2/routeCipher.h . . . 39
7.5.1 Подробное описание . . . 40
7.6 routeCipher.h . . . 41
7.7 Файл lab-3/task1/test.cpp . . . 42
7.7.1 Функции . . . 42
7.7.1.1 assert_exception() . . . 42
7.7.1.2 assert_true() . . . 43
7.7.1.3 main() . . . 43
7.7.1.4 print_section() . . . 43
7.7.1.5 test_constructor() . . . 43

7.7.1.6 test_decrypt() . . . 43
7.7.1.7 test_edge_cases() . . . 43
7.7.1.8 test_encrypt() . . . 43
7.7.1.9 test_integration() . . . 44
7.7.2 Переменные . . . 44
7.7.2.1 passed_tests . . . 44
7.7.2.2 test_passed . . . 44
7.7.2.3 total_tests . . . 44
7.8 Файл lab-3/task2/test.cpp . . . 44
7.8.1 Функции . . . 45
7.8.1.1 assert_exception() . . . 45
7.8.1.2 assert_true() . . . 45
7.8.1.3 main() . . . 45
7.8.1.4 print_section() . . . 45
7.8.1.5 test_constructor() . . . 46
7.8.1.6 test_decrypt() . . . 46
7.8.1.7 test_edge_cases() . . . 46
7.8.1.8 test_encrypt() . . . 46
7.8.1.9 test_integration() . . . 46
7.8.2 Переменные . . . 46
7.8.2.1 passed_tests . . . 46
7.8.2.2 total_tests . . . 46
7.9 Файл lab-3/README.md . . . 46
7.10 Файл README.md . . . 46

Предметный указатель 47

Глава 1

lab-2

Глава 2

lab-4

Глава 3

Иерархический список классов

3.1 Иерархия классов

Иерархия классов.

std::invalid_argument
    route_cipher_error 19
modAlphaCipher 11
routeCipher 20

Глава 4

Алфавитный указатель классов

4.1 Классы

Классы с их кратким описанием.

modAlphaCipher
: Класс для шифрования и расшифрования текста методом Гронсфельда 11

route_cipher_error
: Класс для обработки ошибок маршрутного шифрования 19

routeCipher
: Класс для маршрутного (табличного) шифрования 20

Глава 5

Список файлов

5.1 Файлы

Полный список файлов.

lab-3/task1/modAlphaCipher.cpp
Реализация модуля шифрования методом Гронсфельда 31
lab-3/task1/modAlphaCipher.h 32
lab-3/task1/test.cpp 42
lab-3/task2/routeCipher.cpp
Реализация модуля маршрутного шифрования 38
lab-3/task2/routeCipher.h
Заголовочный файл для модуля маршрутного шифрования 39
lab-3/task2/test.cpp 44

Глава 6

Классы

6.1 Класс modAlphaCipher

Класс для шифрования и расшифрования текста методом Гронсфельда

#include <modAlphaCipher.h>

Открытые члены

- modAlphaCipher ()=delete
  Запрет конструктора без параметров
- modAlphaCipher (const std::wstring &skey)
  Конструктор с установкой ключа
- std::wstring encrypt (const std::wstring &open_text)
  Зашифрование текста
- std::wstring decrypt (const std::wstring &cipher_text)
  Расшифрование текста

Закрытые члены

- std::vector< int > convert (const std::wstring &s)
  Преобразование строки в числовой вектор
- std::wstring convert (const std::vector< int > &v)
  Преобразование числового вектора в строку
- std::wstring getValidKey (const std::wstring &s)
  Валидация и нормализация ключа
- std::wstring getValidOpenText (const std::wstring &s)
  Валидация и нормализация открытого текста
- std::wstring getValidCipherText (const std::wstring &s)
  Валидация зашифрованного текста

Закрытые данные

- std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"

  Алфавит по порядку (33 символа)
- std::map< wchar_t, int > alphaNum

  Ассоциативный массив "символ -> номер".
- std::vector< int > key

  Ключ в числовом виде

6.1.1 Подробное описание

Класс для шифрования и расшифрования текста методом Гронсфельда

Реализует алгоритм шифрования Гронсфельда для русского алфавита. Ключ устанавливается в конструкторе. Длина ключа может быть меньше текста, в этом случае ключ используется циклически.

Алгоритм:

1. Каждой букве алфавита присваивается номер (0-32 для русского)
2. Текст и ключ преобразуются в числовые векторы
3. Выполняется сложение по модулю 33 (размер алфавита)
4. Результат преобразуется обратно в буквы

Заметки

Для расшифрования используется тот же ключ

6.1.2 Конструктор(ы)

6.1.2.1 modAlphaCipher() [1/2]

modAlphaCipher::modAlphaCipher ( ) [delete]

Запрет конструктора без параметров

6.1.2.2 modAlphaCipher() [2/2]

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

Конструктор с установкой ключа

Аргументы

in	skey	Ключ для шифрования (строка русских букв)

Исключения

cipher_error	При недопустимом ключе (см. getValidKey)

Пример использования:

```
modAlphaCipher cipher(L"ПАРОЛЬ");
```

6.1.3 Методы

6.1.3.1 convert() [1/2]

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

Преобразование числового вектора в строку

Аргументы

in	v	Числовой вектор (номера букв)

Возвращает

Строка, составленная из букв алфавита

Исключения

cipher_error	Если номер выходит за границы алфавита

6.1.3.2 convert() [2/2]

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

Преобразование строки в числовой вектор

Аргументы

in	s	Строка для преобразования (только русские буквы)

Возвращает

Вектор целых чисел, где каждое число - номер буквы в алфавите

Исключения

cipher_error	Если символ не найден в алфавите

6.1.3.3 decrypt()

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

Расшифрование текста

Аргументы

in	cipher_text	Зашифрованный текст

Возвращает

Расшифрованный текст (только заглавные русские буквы)

Исключения

cipher_error	При недопустимом тексте (см. getValidCipherText)

Алгоритм:

1. Валидация текста
2. Преобразование в числовой вектор
3. Поэлементное вычитание ключа по модулю 33
4. Преобразование результата в строку

Аргументы

in	cipher_text	Зашифрованный текст

Возвращает

Расшифрованный текст

Исключения

cipher_error	При недопустимом тексте

6.1.3.4 encrypt()

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

Зашифрование текста

Шифрование текста

Аргументы

in	open_text	Открытый текст для шифрования

Возвращает

Зашифрованный текст (только заглавные русские буквы)

Исключения

cipher_error	При недопустимом тексте (см. getValidOpenText)

Алгоритм:

1. Валидация текста
2. Преобразование в числовой вектор
3. Поэлементное сложение с ключом по модулю 33
4. Преобразование результата в строку

Аргументы

in	open_text	Открытый текст

Возвращает

Зашифрованный текст

Исключения

cipher_error	При недопустимом тексте

6.1.3.5 getValidCipherText()

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

Валидация зашифрованного текста

Аргументы

in	s	Зашифрованный текст

Возвращает

Валидированный текст

Исключения

cipher_error	В случаях: • Текст пустой • Текст содержит не-буквы • Текст содержит строчные буквы

Аргументы

in	s	Зашифрованный текст

Возвращает

Валидированный текст

Исключения

cipher_error	При недопустимом тексте

6.1.3.6 getValidKey()

wstring modAlphaCipher::getValidKey (

const std::wstring & s) [private]

Валидация и нормализация ключа

Аргументы

in	s	Ключ в виде строки

Возвращает

Валидированный ключ (все буквы в верхнем регистре)

Исключения

cipher_error	В случаях: • Ключ пустой • Ключ содержит не-буквы • Все символы ключа одинаковы (вырожденный ключ)

Аргументы

in	s	Ключ в виде строки

Возвращает

Валидированный ключ

Исключения

cipher_error	При недопустимом ключе

6.1.3.7 getValidOpenText()

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

Валидация и нормализация открытого текста

Аргументы

in	s	Открытый текст

Возвращает

Валидированный текст (только русские буквы в верхнем регистре)

Удаляет все пробелы, знаки препинания, цифры

Исключения

cipher_error	Если после очистки текст пустой

Аргументы

in	s	Открытый текст

Возвращает

Валидированный текст

Исключения

cipher_error	При недопустимом тексте

6.1.4 Данные класса

6.1.4.1 alphaNum

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

Ассоциативный массив "символ -> номер".

6.1.4.2 key

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

Ключ в числовом виде

6.1.4.3 numAlpha

std::wstring modAlphaCipher::numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ" [private]

Алфавит по порядку (33 символа)

Объявления и описания членов классов находятся в файлах:

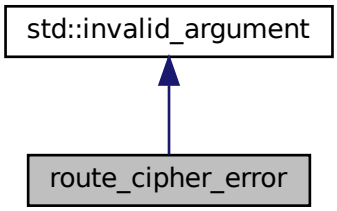
- lab-3/task1/modAlphaCipher.h
- lab-3/task1/modAlphaCipher.cpp

6.2 Класс route_cipher_error

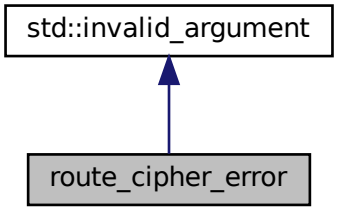
Класс для обработки ошибок маршрутного шифрования

#include <routeCipher.h>

Граф наследования:route_cipher_error:



Граф связей класса route_cipher_error:



Открытые члены

- route_cipher_error (const std::string &what_arg)
  Конструктор с строковым параметром
- route_cipher_error (const char *what_arg)
  Конструктор с C-строкой

6.2.1 Подробное описание

Класс для обработки ошибок маршрутного шифрования

Наследуется от std::invalid_argument. Используется для генерации исключительных ситуаций при недопустимых входных данных.

См. также

std::invalid_argument

6.2.2 Конструктор(ы)

6.2.2.1 route_cipher_error() [1/2]

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

Конструктор с строковым параметром

Аргументы

in	what_arg	Сообщение об ошибке в виде объекта std::string

6.2.2.2 route_cipher_error() [2/2]

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

Конструктор с C-строкой

Аргументы

in	what_arg	Сообщение об ошибке в виде C-строки

Объявления и описания членов класса находятся в файле:

- lab-3/task2/routeCipher.h

6.3 Класс routeCipher

Класс для маршрутного (табличного) шифрования

#include <modAlphaCipher.h>

Открытые члены

- routeCipher ()=delete
- routeCipher (int cols)

  Конструктор класса routeCipher.
- std::wstring encrypt (const std::wstring &text)
- std::wstring decrypt (const std::wstring &text)
- routeCipher ()=delete

Удаленный конструктор по умолчанию
- routeCipher (int cols)

  Конструктор с установкой количества столбцов
- std::wstring encrypt (const std::wstring &text)

  Шифрование текста
- std::wstring decrypt (const std::wstring &text)

  Расшифрование текста

Закрытые члены

- std::wstring prepareText (const std::wstring &text)
- std::vector< std::vector< wchar_t > > createTable (const std::wstring &text, int cols)
- std::wstring readEncrypted (const std::vector< std::vector< wchar_t > > &table, int cols)
- std::wstring readDecrypted (const std::vector< std::vector< wchar_t > > &table, int cols)
- void validateColumns (int cols)

  Валидация количества столбцов
- std::wstring getValidOpenText (const std::wstring &s)
- std::wstring getValidCipherText (const std::wstring &s)
- std::wstring prepareText (const std::wstring &text)

  Подготовка текста к обработке
- std::vector< std::vector< wchar_t > > createTable (const std::wstring &text, int cols)

  Создание таблицы для шифрования
- std::wstring readEncrypted (const std::vector< std::vector< wchar_t > > &table, int cols)

  Чтение таблицы в режиме шифрования
- std::wstring readDecrypted (const std::vector< std::vector< wchar_t > > &table, int cols)

  Чтение таблицы в режиме расшифрования
- void validateColumns (int cols)

  Валидация количества столбцов
- std::wstring getValidOpenText (const std::wstring &s)

  Валидация открытого текста
- std::wstring getValidCipherText (const std::wstring &s)

  Валидация зашифрованного текста

Закрытые данные

- int columns

  Количество столбцов в таблице

6.3.1 Подробное описание

Класс для маршрутного (табличного) шифрования

Реализует алгоритм маршрутного шифрования, который заключается в заполнении таблицы текстом по строкам и чтении его по столбцам в определенном порядке.

Принцип работы:

1. Текст записывается в таблицу построчно
2. Пустые ячейки заполняются пробелами
3. Для шифрования: чтение таблицы по столбцам снизу вверх справа налево
4. Для расшифрования: заполнение таблицы по столбцам и чтение построчно

Пример шифрования:

```
Текст: "ПРИВЕТМИР", Колонки: 3
Таблица:
| П | Р | И |
| В | Е | Т |
| М | И | Р |
Чтение для шифрования (по столбцам справа налево снизу вверх):
Столбец 3: Р, Т, Р → "РТР"
Столбец 2: И, Е, И → "ИЕИ"
Столбец 1: П, В, М → "ПВМ"
Результат: "РТРИЕИПВМ"
```

Заметки

Алгоритм не является криптостойким и предназначен для учебных целей

6.3.2 Конструктор(ы)

6.3.2.1 routeCipher() [1/4]

routeCipher::routeCipher ( ) [delete]

6.3.2.2 routeCipher() [2/4]

routeCipher::routeCipher (
int cols)

Конструктор класса routeCipher.

Аргументы

in	cols	Количество столбцов в таблице

Исключения

route_cipher_error	При недопустимом количестве столбцов

6.3.2.3 routeCipher() [3/4]

routeCipher::routeCipher ( ) [delete]

Удаленный конструктор по умолчанию

6.3.2.4 routeCipher() [4/4]

routeCipher::routeCipher (
int cols)

Конструктор с установкой количества столбцов

Аргументы

in	cols	Количество столбцов для таблицы

Исключения

route_cipher_error	При недопустимом количестве столбцов

6.3.3 Методы

6.3.3.1 createTable() [1/2]

std::vector< std::vector< wchar_t > > routeCipher::createTable (
const std::wstring & text,
int cols) [private]

6.3.3.2 createTable() [2/2]

std::vector< std::vector< wchar_t > > routeCipher::createTable (
const std::wstring & text,
int cols) [private]

Создание таблицы для шифрования

Аргументы

in	text	Текст для размещения в таблице
in	cols	Количество столбцов

Возвращает

Двумерный вектор символов (таблица)

Заметки

Пустые ячейки заполняются пробелами

6.3.3.3 decrypt() [1/2]

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

6.3.3.4 decrypt() [2/2]

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

Расшифрование текста

Аргументы

in	text	Зашифрованный текст

Возвращает

Расшифрованный текст

Исключения

route_cipher_error	При недопустимом тексте

6.3.3.5 encrypt() [1/2]

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

6.3.3.6 encrypt() [2/2]

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

Шифрование текста

Аргументы

in	text	Текст для шифрования

Возвращает

Зашифрованный текст

Исключения

route_cipher_error	При недопустимом тексте

6.3.3.7 getValidCipherText() [1/2]

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

6.3.3.8 getValidCipherText() [2/2]

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

Валидация зашифрованного текста

Аргументы

in	s	Зашифрованный текст

Возвращает

Валидированный текст

Исключения

route_cipher_error	При недопустимом тексте

6.3.3.9 getValidOpenText() [1/2]

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

6.3.3.10 getValidOpenText() [2/2]

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

Валидация открытого текста

Аргументы

in	s	Открытый текст

Возвращает

Валидированный текст

Исключения

route_cipher_error	При недопустимом тексте

6.3.3.11 prepareText() [1/2]

std::wstring routeCipher::prepareText (
const std::wstring & text) [private]

6.3.3.12 prepareText() [2/2]

std::wstring routeCipher::prepareText (
const std::wstring & text) [private]

Подготовка текста к обработке

Аргументы

in	text	Исходный текст

Возвращает

Текст без пробелов в верхнем регистре

Удаляет пробелы и приводит все буквы к верхнему регистру

6.3.3.13 readDecrypted() [1/2]

std::wstring routeCipher::readDecrypted (
const std::vector< std::vector< wchar_t > > & table,
int cols) [private]

6.3.3.14 readDecrypted() [2/2]

std::wstring routeCipher::readDecrypted (
const std::vector< std::vector< wchar_t > > & table,
int cols) [private]

Чтение таблицы в режиме расшифрования

Аргументы

in	table	Таблица символов
in	cols	Количество столбцов

Возвращает

Расшифрованная строка

Читает таблицу построчно слева направо

6.3.3.15 readEncrypted() [1/2]

std::wstring routeCipher::readEncrypted (
const std::vector< std::vector< wchar_t > > & table,
int cols) [private]

6.3.3.16 readEncrypted() [2/2]

std::wstring routeCipher::readEncrypted (
const std::vector< std::vector< wchar_t > > & table,
int cols) [private]

Чтение таблицы в режиме шифрования

Аргументы

in	table	Таблица символов
in	cols	Количество столбцов

Возвращает

Зашифрованная строка

Читает таблицу по столбцам справа налево снизу вверх

6.3.3.17 validateColumns() [1/2]

void routeCipher::validateColumns (
int cols) [private]

Валидация количества столбцов

Аргументы

in	cols	Количество столбцов для проверки

Исключения

route_cipher_error	При недопустимом количестве столбцов

6.3.3.18 validateColumns() [2/2]

void routeCipher::validateColumns (
int cols) [private]

Валидация количества столбцов

Аргументы

in	cols	Количество столбцов для проверки

Исключения

route_cipher_error	При недопустимом количестве столбцов

6.3.4 Данные класса

6.3.4.1 columns

int routeCipher::columns [private]

Количество столбцов в таблице

Объявления и описания членов классов находятся в файлах:

- lab-3/task1/modAlphaCipher.h
- lab-3/task2/routeCipher.h
- lab-3/task2/routeCipher.cpp

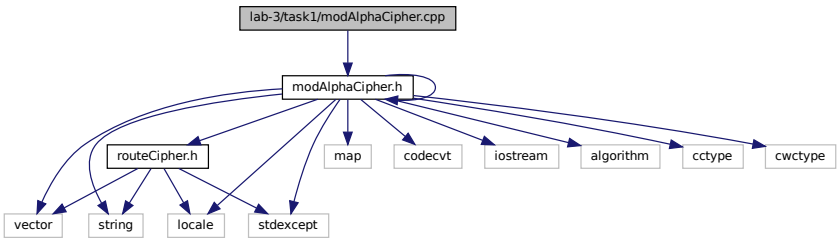
Глава 7

Файлы

7.1 Файл lab-3/task1/modAlphaCipher.cpp

Реализация модуля шифрования методом Гронсфельда

#include "modAlphaCipher.h"
Граф включаемых заголовочных файлов для modAlphaCipher.cpp:



7.1.1 Подробное описание

Реализация модуля шифрования методом Гронсфельда

Автор

Павлова В.М.

Версия

1.0

Дата

25.12.2025

Данный файл содержит реализацию методов класса modAlphaCipher, объявленного в файле mod↩ AlphaCipher.h.

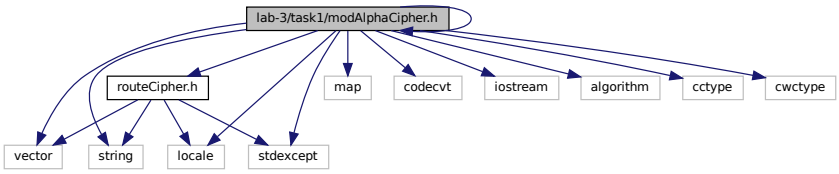
См. также

modAlphaCipher.h

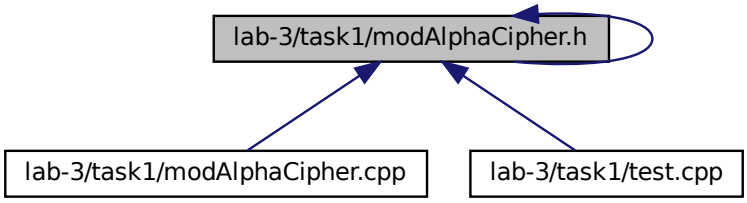
7.2 Файл lab-3/task1/modAlphaCipher.h

```
#include "modAlphaCipher.h"
#include <vector>
#include <string>
#include <map>
#include <locale>
#include <codecvt>
#include <stdexcept>
#include <iostream>
#include "routeCipher.h"
#include <algorithm>
#include <cctype>
#include <cwctype>
```

Граф включаемых заголовочных файлов для modAlphaCipher.h:



Граф файлов, в которые включается этот файл:



Классы

- class modAlphaCipher

  Класс для шифрования и расшифрования текста методом Гронсфельда
- class routeCipher

  Класс для маршрутного (табличного) шифрования

Переменные

- смотри
- у меня в лабе было задания и они в разных папках и при этом не было ни каких main BOT task1
- modAlphaCipher h
- routeCipher cpp
- columns = cols

7.2.1 Переменные

7.2.1.1 смотри

смотри

7.2.1.2 columns

columns = cols

7.2.1.3 cpp

routeCipher cpp

7.2.1.4 h

routeCipher h

7.2.1.5 task1

у меня в лабе было задания и они в разных папках и при этом не было ни каких main BOT task1

7.3 modAlphaCipher.h

См. документацию.

```
1 смотри, у меня в 3 лабе было 2 задания и они в разных папках и при этом не было ни каких main
2 BOT
3 task1:
4
5 modAlphaCipher.cpp:
15 #include "modAlphaCipher.h"
16
17 using namespace std;
18
19 // Конструктор
20 modAlphaCipher::modAlphaCipher(const wstring& skey) {
21     // Инициализация ассоциативного массива alphaNum
22     for (size_t i = 0; i < numAlpha.size(); i++) {
23         alphaNum[numAlpha[i]] = i;
24     }
25     // Валидация и преобразование ключа
26     key = convert(getValidKey(skey));
27 }
28
29 // Преобразование строки в вектор
30 vector<int> modAlphaCipher::convert(const wstring& s) {
31     vector<int> result;
32     for (auto c : s) {
33         result.push_back(alphaNum[c]);
34     }
35     return result;
36 }
37
38 // Преобразование вектора в строку
39 wstring modAlphaCipher::convert(const vector<int>& v) {
40     wstring result;
41     for (auto i : v) {
42         result.push_back(numAlpha[i]);
43     }
44     return result;
45 }
46
47 // Валидация ключа
48 wstring modAlphaCipher::getValidKey(const wstring& s) {
49     if (s.empty()) {
50         throw cipher_error("Пустой ключ");
51     }
52
53     wstring tmp(s);
54     // Проверка на не-буквы и перевод в верхний регистр
55     for (auto& c : tmp) {
56         if (!isalpha(c, locale("ru_RU.UTF-8"))) {
57             wstring_convert<codecvt_utf8<wchar_t>> converter;
58             throw cipher_error("Недопустимый символ в ключе: " + converter.to_bytes(wstring(1, c)));
59         }
60         if (islower(c, locale("ru_RU.UTF-8"))) {
61             c = toupper(c, locale("ru_RU.UTF-8"));
62         }
63     }
64
65     // Проверка на вырожденный ключ (все символы одинаковы)
66     bool allSame = true;
67     for (size_t i = 1; i < tmp.size(); i++) {
68         if (tmp[i] != tmp[0]) {
69             allSame = false;
70             break;
71         }
72     }
73     if (allSame) {
74         throw cipher_error("Вырожденный ключ (все символы одинаковы)");
75     }
76
77     return tmp;
78 }
79
80 // Валидация открытого текста
81 wstring modAlphaCipher::getValidOpenText(const wstring& s) {
82     wstring tmp;
83     for (auto c : s) {
84         if (isalpha(c, locale("ru_RU.UTF-8"))) {
85             if (islower(c, locale("ru_RU.UTF-8"))) {
86                 tmp.push_back(toupper(c, locale("ru_RU.UTF-8")));
87             } else {
88                 tmp.push_back(c);
89             }
90         }
91     }
```

```
92
93     if (tmp.empty()) {
94         throw cipher_error("Пустой открытый текст после очистки");
95     }
96
97     return tmp;
98 }
99
100 // Валидация зашифрованного текста
101 wstring modAlphaCipher::getValidCipherText(const wstring& s) {
102     if (s.empty()) {
103         throw cipher_error("Пустой зашифрованный текст");
104     }
105
106     for (auto c : s) {
107         if (!isalpha(c, locale("ru_RU.UTF-8")) || islower(c, locale("ru_RU.UTF-8"))) {
108             wstring_convert<codecvt_utf8<wchar_t» converter;
109             throw cipher_error("Недопустимый символ в зашифрованном тексте: " + converter.to_bytes(wstring(1, c)));
110         }
111     }
112
113     return s;
114 }
115
116 // Шифрование
117 wstring modAlphaCipher::encrypt(const wstring& open_text) {
118     wstring valid_text = getValidOpenText(open_text);
119     vector<int> work = convert(valid_text);
120
121     for (size_t i = 0; i < work.size(); i++) {
122         work[i] = (work[i] + key[i % key.size()]) % numAlpha.size();
123     }
124
125     return convert(work);
126 }
127
128 // Расшифрование
129 wstring modAlphaCipher::decrypt(const wstring& cipher_text) {
130     wstring valid_text = getValidCipherText(cipher_text);
131     vector<int> work = convert(valid_text);
132
133     for (size_t i = 0; i < work.size(); i++) {
134         work[i] = (work[i] + numAlpha.size() - key[i % key.size()]) % numAlpha.size();
135     }
136
137     return convert(work);
138 }
139
140 modAlphaCipher.h:
141
152 #pragma once
153 #include <vector>
154 #include <string>
155 #include <map>
156 #include <locale>
157 #include <codecvt>
158 #include <stdexcept>
159 #include <iostream>
160
167 class cipher_error: public std::invalid_argument {
168 public:
173     explicit cipher_error (const std::string& what_arg):
174         std::invalid_argument(what_arg) {}
175
180     explicit cipher_error (const char* what_arg):
181         std::invalid_argument(what_arg) {}
182 };
183
199 class modAlphaCipher {
200 private:
201     std::wstring numAlpha = L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";
202     std::map <wchar_t,int> alphaNum;
203     std::vector <int> key;
204
211     std::vector<int> convert(const std::wstring& s);
212
219     std::wstring convert(const std::vector<int>& v);
220
230     std::wstring getValidKey(const std::wstring& s);
231
239     std::wstring getValidOpenText(const std::wstring& s);
240
250     std::wstring getValidCipherText(const std::wstring& s);
251
252 public:
253     modAlphaCipher() = delete;
254
```

```
265     modAlphaCipher(const std::wstring& skey);
266
279     std::wstring encrypt(const std::wstring& open_text);
280
293     std::wstring decrypt(const std::wstring& cipher_text);
294 };
295
296 routeCipher.cpp:
297
298 #include "routeCipher.h"
299 #include <algorithm>
300 #include <cctype>
301 #include <locale>
302 #include <cwctype>
303
304 routeCipher::routeCipher(int cols)
305 {
306     validateColumns(cols);
307     columns = cols;
308 }
309
310 // Валидация количества столбцов
311 void routeCipher::validateColumns(int cols)
312 {
313     if (cols <= 0) {
314         throw route_cipher_error("Number of columns must be positive");
315     }
316     if (cols > 100) { // Реалистичное ограничение
317         throw route_cipher_error("Number of columns is too large");
318     }
319 }
320
321 // Валидация открытого текста
322 std::wstring routeCipher::getValidOpenText(const std::wstring& s)
323 {
324     if (s.empty()) {
325         throw route_cipher_error("Empty open text");
326     }
327
328     std::wstring result;
329     std::locale loc("ru_RU.UTF-8");
330
331     for (wchar_t c : s) {
332         if (isalpha(c, loc) || c == L' ') {
333             if (isalpha(c, loc)) {
334                 c = toupper(c, loc);
335             }
336             result += c;
337         }
338         // Игнорируем другие символы (цифры, знаки препинания)
339     }
340
341     // Удаляем пробелы для шифрования
342     std::wstring textWithoutSpaces;
343     for (wchar_t c : result) {
344         if (c != L' ') {
345             textWithoutSpaces += c;
346         }
347     }
348
349     if (textWithoutSpaces.empty()) {
350         throw route_cipher_error("Open text contains no valid letters");
351     }
352
353     return textWithoutSpaces;
354 }
355
356 // Валидация зашифрованного текста
357 std::wstring routeCipher::getValidCipherText(const std::wstring& s)
358 {
359     if (s.empty()) {
360         throw route_cipher_error("Empty cipher text");
361     }
362
363     std::locale loc("ru_RU.UTF-8");
364     for (wchar_t c : s) {
365         if (!isalpha(c, loc)) {
366             throw route_cipher_error("Cipher text must contain only letters");
367         }
368         if (!isupper(c, loc)) {
369             throw route_cipher_error("Cipher text must be in uppercase");
370         }
371     }
372
373     return s;
374 }
375
```

```
376 std::wstring routeCipher::prepareText(const std::wstring& text)
377 {
378     std::wstring result;
379     std::locale loc("ru_RU.UTF-8");
380 
381     for (wchar_t c : text) {
382         if (c != L' ') {
383             c = toupper(c, loc);
384             result += c;
385         }
386     }
387     return result;
388 }
389 
390 std::vector<std::vector<wchar_t» routeCipher::createTable(const std::wstring& text, int cols)
391 {
392     int length = text.length();
393     int rows = (length + cols - 1) / cols;
394     std::vector<std::vector<wchar_t» table(rows, std::vector<wchar_t>(cols, L' '));
395 
396     int index = 0;
397     for (int i = 0; i < rows; i++) {
398         for (int j = 0; j < cols; j++) {
399             if (index < length) {
400                 table[i][j] = text[index++];
401             }
402         }
403     }
404     return table;
405 }
406 
407 std::wstring routeCipher::readEncrypted(const std::vector<std::vector<wchar_t»& table, int cols)
408 {
409     int rows = table.size();
410     std::wstring result;
411     for (int j = cols - 1; j >= 0; j--) {
412         for (int i = 0; i < rows; i++) {
413             if (table[i][j] != L' ') {
414                 result += table[i][j];
415             }
416         }
417     }
418     return result;
419 }
420 
421 std::wstring routeCipher::readDecrypted(const std::vector<std::vector<wchar_t»& table, int cols)
422 {
423     int rows = table.size();
424     std::wstring result;
425     for (int i = 0; i < rows; i++) {
426         for (int j = 0; j < cols; j++) {
427             if (table[i][j] != L' ') {
428                 result += table[i][j];
429             }
430         }
431     }
432     return result;
433 }
434 
435 std::wstring routeCipher::encrypt(const std::wstring& text)
436 {
437     std::wstring prepared = getValidOpenText(text);
438     auto table = createTable(prepared, columns);
439     return readEncrypted(table, columns);
440 }
441 
442 std::wstring routeCipher::decrypt(const std::wstring& text)
443 {
444     std::wstring prepared = getValidCipherText(text);
445     int length = prepared.length();
446 
447     if (length == 0) {
448         throw route_cipher_error("Cipher text is empty after validation");
449     }
450 
451     int rows = (length + columns - 1) / columns;
452 
453     int extras = length % columns;
454     std::vector<int> heights(columns, rows);
455     if (extras != 0) {
456         for (int j = 0; j < columns; ++j) {
457             heights[j] = (j < extras) ? rows : (rows - 1);
458         }
459     }
460 
461     std::vector<std::vector<wchar_t» table(rows, std::vector<wchar_t>(columns, L' '));
462     int index = 0;
```

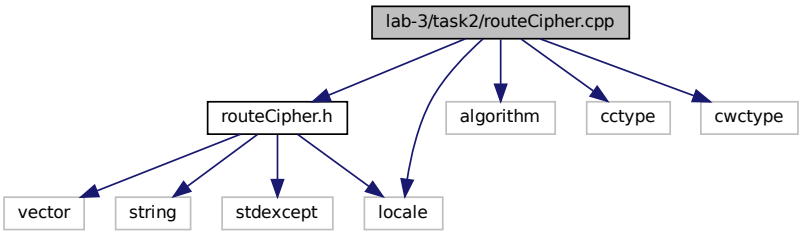
```
463
464     for (int j = columns - 1; j >= 0; j--) {
465         int h = heights[j];
466         for (int i = 0; i < h; i++) {
467             if (index < length) {
468                 table[i][j] = prepared[index++];
469             }
470         }
471     }
472     return readDecrypted(table, columns);
473 }
474
475
476 routeCipher.h:
477
478 #pragma once
479 #include <vector>
480 #include <string>
481 #include <stdexcept>
482 #include <locale>
483
484 class route_cipher_error : public std::invalid_argument {
485 public:
486     explicit route_cipher_error(const std::string& what_arg) :
487         std::invalid_argument(what_arg) {}
488     explicit route_cipher_error(const char* what_arg) :
489         std::invalid_argument(what_arg) {}
490 };
491
492 class routeCipher
493 {
494 private:
495     int columns;
496
497     std::wstring prepareText(const std::wstring& text);
498     std::vector<std::vector<wchar_t» createTable(const std::wstring& text, int cols);
499     std::wstring readEncrypted(const std::vector<std::vector<wchar_t»& table, int cols);
500     std::wstring readDecrypted(const std::vector<std::vector<wchar_t»& table, int cols);
501
502     // Методы валидации
503     void validateColumns(int cols);
504     std::wstring getValidOpenText(const std::wstring& s);
505     std::wstring getValidCipherText(const std::wstring& s);
506
507 public:
508     routeCipher() = delete;
509     routeCipher(int cols);
510
511     std::wstring encrypt(const std::wstring& text);
512     std::wstring decrypt(const std::wstring& text);
513 };
```

7.4 Файл lab-3/task2/routeCipher.cpp

Реализация модуля маршрутного шифрования

```
#include "routeCipher.h"
#include <algorithm>
#include <cctype>
#include <locale>
#include <cwctype>
```

Граф включаемых заголовочных файлов для routeCipher.cpp:



7.4.1 Подробное описание

Реализация модуля маршрутного шифрования

Автор

Павлова В.М.

Версия

1.0

Дата

25.12.2025

Данный файл содержит реализацию методов класса routeCipher, объявленного в файле route↩ Cipher.h.

См. также

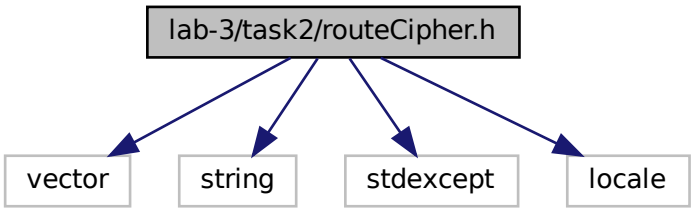
routeCipher.h

7.5 Файл lab-3/task2/routeCipher.h

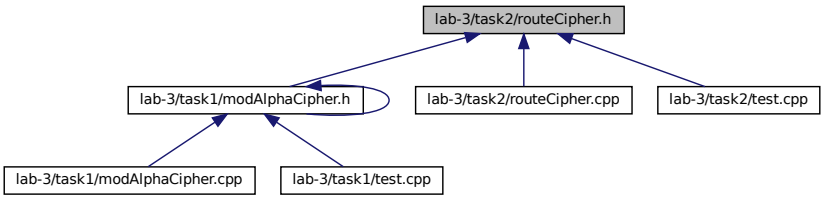
Заголовочный файл для модуля маршрутного шифрования

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

#include <locale>
Граф включаемых заголовочных файлов для routeCipher.h:



Граф файлов, в которые включается этот файл:



Классы

- class route_cipher_error

  Класс для обработки ошибок маршрутного шифрования
- class routeCipher

  Класс для маршрутного (табличного) шифрования

7.5.1 Подробное описание

Заголовочный файл для модуля маршрутного шифрования

Автор

Павлова В.М.

Версия

1.0

Дата

25.12.2025

Данный файл содержит объявление класса routeCipher, реализующего алгоритм маршрутного (табличного) шифрования для русского алфавита.

Предупреждения

Реализация предназначена только для русского языка

Авторство

7.6 routeCipher.h

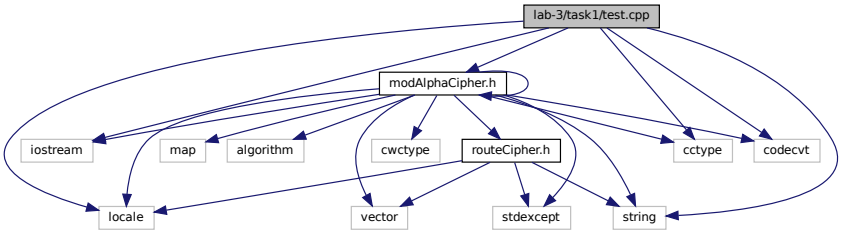
См. документацию.

```
1
13 #pragma once
14
15 #include <vector>
16 #include <string>
17 #include <stdexcept>
18 #include <locale>
19
27 class route_cipher_error : public std::invalid_argument {
28 public:
33     explicit route_cipher_error(const std::string& what_arg) :
34         std::invalid_argument(what_arg) {}
35
40     explicit route_cipher_error(const char* what_arg) :
41         std::invalid_argument(what_arg) {}
42 };
43
74 class routeCipher {
75 private:
76     int columns;
77
84     std::wstring prepareText(const std::wstring& text);
85
93     std::vector<std::vector<wchar_t>> createTable(const std::wstring& text, int cols);
94
102     std::wstring readEncrypted(const std::vector<std::vector<wchar_t>>& table, int cols);
103
111     std::wstring readDecrypted(const std::vector<std::vector<wchar_t>>& table, int cols);
112
118     void validateColumns(int cols);
119
126     std::wstring getValidOpenText(const std::wstring& s);
127
134     std::wstring getValidCipherText(const std::wstring& s);
135
136 public:
140     routeCipher() = delete;
141
147     routeCipher(int cols);
148
155     std::wstring encrypt(const std::wstring& text);
156
163     std::wstring decrypt(const std::wstring& text);
164 };
```

7.7 Файл lab-3/task1/test.cpp

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

Граф включаемых заголовочных файлов для test.cpp:



Функции

- void assert_true (bool condition, const string &message)
- void assert_exception (void(*func)(), const string &message)
- void print_section (const string &section_name)
- void test_constructor ()
- void test_encrypt ()
- void test_decrypt ()
- void test_edge_cases ()
- void test_integration ()
- int main ()

Переменные

- bool test_passed = true
- int total_tests = 0
- int passed_tests = 0

7.7.1 Функции

7.7.1.1 assert_exception()

```
void assert_exception (
                void(*)() func,
                const string & message )
```

7.7.1.2 assert_true()

```
void assert_true (
        bool condition,
        const string & message )
```

7.7.1.3 main()

```
int main ( )
```

7.7.1.4 print_section()

```
void print_section (
        const string & section_name )
```

7.7.1.5 test_constructor()

```
void test_constructor ( )
```

7.7.1.6 test_decrypt()

```
void test_decrypt ( )
```

7.7.1.7 test_edge_cases()

```
void test_edge_cases ( )
```

7.7.1.8 test_encrypt()

```
void test_encrypt ( )
```

7.7.1.9 test_integration()

void test_integration ( )

7.7.2 Переменные

7.7.2.1 passed_tests

int passed_tests = 0

7.7.2.2 test_passed

bool test_passed = true

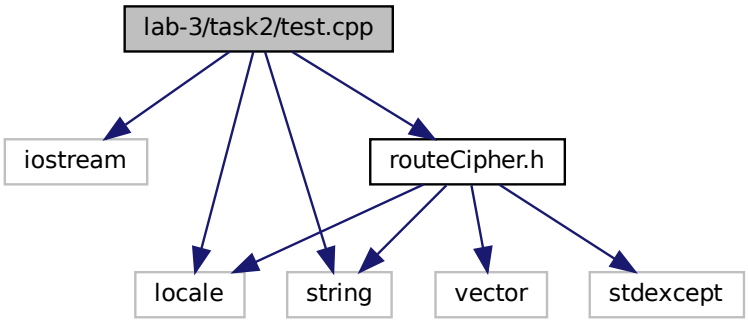
7.7.2.3 total_tests

int total_tests = 0

7.8 Файл lab-3/task2/test.cpp

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

Граф включаемых заголовочных файлов для test.cpp:



Функции

- void assert_true (bool condition, const string &message)
- void assert_exception (void(*func)(), const string &message)
- void print_section (const string &section_name)
- void test_constructor ()
- void test_encrypt ()
- void test_decrypt ()
- void test_edge_cases ()
- void test_integration ()
- int main ()

Переменные

- int total_tests = 0
- int passed_tests = 0

7.8.1 Функции

7.8.1.1 assert_exception()

```
void assert_exception (
                void(*)() func,
                const string & message )
```

7.8.1.2 assert_true()

```
void assert_true (
                bool condition,
                const string & message )
```

7.8.1.3 main()

```
int main ( )
```

7.8.1.4 print_section()

```
void print_section (
                const string & section_name )
```

7.8.1.5 test_constructor()

void test_constructor ( )

7.8.1.6 test_decrypt()

void test_decrypt ( )

7.8.1.7 test_edge_cases()

void test_edge_cases ( )

7.8.1.8 test_encrypt()

void test_encrypt ( )

7.8.1.9 test_integration()

void test_integration ( )

7.8.2 Переменные

7.8.2.1 passed_tests

int passed_tests = 0

7.8.2.2 total_tests

int total_tests = 0

7.9 Файл lab-3/README.md

7.10 Файл README.md

Предметный указатель

alphaNum
  modAlphaCipher, 18
assert_exception
  test.cpp, 42, 45
assert_true
  test.cpp, 42, 45

columns
  modAlphaCipher.h, 33
  routeCipher, 29
convert
  modAlphaCipher, 13
cpp
  modAlphaCipher.h, 33
createTable
  routeCipher, 23

decrypt
  modAlphaCipher, 14
  routeCipher, 24

encrypt
  modAlphaCipher, 15
  routeCipher, 24

getValidCipherText
  modAlphaCipher, 16
  routeCipher, 26
getValidKey
  modAlphaCipher, 16
getValidOpenText
  modAlphaCipher, 17
  routeCipher, 26

h
  modAlphaCipher.h, 33

key
  modAlphaCipher, 18

lab-3/README.md, 46
lab-3/task1/modAlphaCipher.cpp, 31
lab-3/task1/modAlphaCipher.h, 32, 34
lab-3/task1/test.cpp, 42
lab-3/task2/routeCipher.cpp, 38
lab-3/task2/routeCipher.h, 39, 41
lab-3/task2/test.cpp, 44

main
  test.cpp, 43, 45
modAlphaCipher, 11
  alphaNum, 18
  convert, 13
  decrypt, 14
  encrypt, 15
  getValidCipherText, 16
  getValidKey, 16
  getValidOpenText, 17
  key, 18
  modAlphaCipher, 12
  numAlpha, 18
modAlphaCipher.h
  columns, 33
  cpp, 33
  h, 33
  task1, 33
  смотри, 33

numAlpha
  modAlphaCipher, 18

passed_tests
  test.cpp, 44, 46
prepareText
  routeCipher, 27
print_section
  test.cpp, 43, 45

readDecrypted
  routeCipher, 27, 28
readEncrypted
  routeCipher, 28
README.md, 46
route_cipher_error, 19
  route_cipher_error, 20
routeCipher, 20
  columns, 29
  createTable, 23
  decrypt, 24
  encrypt, 24
  getValidCipherText, 26
  getValidOpenText, 26
  prepareText, 27
  readDecrypted, 27, 28
  readEncrypted, 28
  routeCipher, 22, 23
  validateColumns, 29

task1
  modAlphaCipher.h, 33
test.cpp

assert_exception, 42, 45
assert_true, 42, 45
main, 43, 45
passed_tests, 44, 46
print_section, 43, 45
test_constructor, 43, 45
test_decrypt, 43, 46
test_edge_cases, 43, 46
test_encrypt, 43, 46
test_integration, 43, 46
test_passed, 44
total_tests, 44, 46
test_constructor
test.cpp, 43, 45
test_decrypt
test.cpp, 43, 46
test_edge_cases
test.cpp, 43, 46
test_encrypt
test.cpp, 43, 46
test_integration
test.cpp, 43, 46
test_passed
test.cpp, 44
total_tests
test.cpp, 44, 46

validateColumns
routeCipher, 29

смотри
modAlphaCipher.h, 33