

Tabl
1.0

Создано системой Doxygen 1.9.8

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	7
4.1.1 Подробное описание	8
4.1.2 Конструктор(ы)	8
4.1.2.1 Cipher() [1/2]	8
4.1.2.2 Cipher() [2/2]	8
4.1.3 Методы	9
4.1.3.1 encryption()	9
4.1.3.2 getValidCipherText()	10
4.1.3.3 getValidKey()	11
4.1.3.4 getValidOpenText()	12
4.1.3.5 transcript()	13
4.2 Класс cipher_error	15
4.2.1 Подробное описание	16
4.2.2 Конструктор(ы)	16
4.2.2.1 cipher_error() [1/2]	16
4.2.2.2 cipher_error() [2/2]	16
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 main()	19
5.2 Файл TableRouteCipher.cpp	19
5.2.1 Подробное описание	20
5.3 Файл TableRouteCipher.h	20
5.3.1 Подробное описание	21
5.4 TableRouteCipher.h	21
Предметный указатель	23

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

Cipher	7
std::invalid_argument	
cipher_error	15
cipher_error	15

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

Cipher	Класс для шифрования методом маршрутной перестановки	7
cipher_error	Класс исключений для ошибок шифрования	15

Глава 3

Список файлов

3.1 Файлы

Полный список документированных файлов.

main.cpp	Демонстрационная программа для тестирования шифра маршрутной перестановки	17
TableRouteCipher.cpp	Реализация класса шифрования методом маршрутной перестановки	19
TableRouteCipher.h	Заголовочный файл для класса шифрования методом маршрутной перестановки	20

Глава 4

Классы

4.1 Класс Cipher

Класс для шифрования методом маршрутной перестановки

```
#include <TableRouteCipher.h>
```

Открытые члены

- `Cipher` `()=delete`
Удаленный конструктор по умолчанию
- `Cipher` `(int skey, string text)`
Конструктор с установкой ключа шифрования
- `string encryption` `(string &text)`
Шифрует открытый текст методом маршрутной перестановки
- `string transcript` `(string &text, string &open_text)`
Расшифровывает зашифрованный текст

Закрытые члены

- `int getValidKey` `(int key, const string &Text)`
Проверяет валидность ключа шифрования
- `string getValidOpenText` `(string &s)`
Проверяет и нормализует открытый текст
- `string getValidCipherText` `(string &s, string &open_text)`
Проверяет валидность зашифрованного текста

Закрытые данные

- `int key`
Ключ шифрования - количество столбцов в таблице

4.1.1 Подробное описание

Класс для шифрования методом маршрутной перестановки

Реализует табличное шифрование, где текст записывается в таблицу по строкам и считывается по столбцам в обратном порядке.

Алгоритм работы:

- Текст разбивается на строки таблицы (количество строк = длина текста / ключ)
- Запись выполняется слева направо, сверху вниз
- Считывание выполняется снизу вверх, справа налево
- Для неполных блоков добавляются заполнители '0'

Особенности:

- Поддержка только английского алфавита
- Автоматическое удаление пробелов при шифровании
- Строгая валидация входных данных
- Использование динамических массивов для таблицы

4.1.2 Конструктор(ы)

4.1.2.1 Cipher() [1/2]

```
Cipher::Cipher ( ) [delete]
```

Удаленный конструктор по умолчанию

Класс требует обязательной установки ключа и текста при создании

4.1.2.2 Cipher() [2/2]

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

Конструктор с установкой ключа шифрования

Аргументы

skey	Ключ шифрования (количество столбцов)
text	Текст для инициализации (используется для проверки ключа)

Исключения

<code>cipher_error</code>	Если ключ невалиден
---------------------------	---------------------

Пример использования:

```
Cipher cipher(4, "HELLO"); // Создание шифратора с 4 столбцами
```

Аргументы

skey	Ключ шифрования (количество столбцов)
text	Текст для инициализации (используется для проверки ключа)

Исключения

<code>cipher_error</code>	Если ключ невалиден
---------------------------	---------------------

Инициализирует объект шифратора с проверкой валидности ключа относительно предоставленного текста.

См. также

[getValidKey](#)

4.1.3 Методы

4.1.3.1 encryption()

```
string Cipher::encryption (  
    string & text )
```

Шифрует открытый текст методом маршрутной перестановки

Аргументы

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

Возвращает

Зашифрованная строка

Исключения

<code>cipher_error</code>	Если текст пуст или содержит недопустимые символы
---------------------------	---

Алгоритм шифрования:

1. Валидация и нормализация текста

2. Создание таблицы $\text{key} \times (\text{n}/\text{key})$ столбцов
3. Запись текста в таблицу по строкам
4. Считывание текста по столбцам справа налево
5. Удаление символов-заполнителей '0'

Пример:

```
string encrypted = cipher.encryption("HELLO WORLD");
```

Аргументы

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

Возвращает

Зашифрованная строка

Исключения

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

Выполняет следующие шаги:

1. Валидация и подготовка текста
2. Создание динамической таблицы для записи текста
3. Запись текста в таблицу по строкам (с заполнением '0')
4. Считывание текста из таблицы по столбцам справа налево
5. Освобождение памяти таблицы
6. Удаление символов-заполнителей '0'

Заметки

Использует динамическое выделение памяти для таблицы
Пустые ячейки заполняются символом '0'

См. также

[getValidOpenText](#)

4.1.3.2 getValidCipherText()

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

Проверяет валидность зашифрованного текста

Аргументы

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

Возвращает

Валидный зашифрованный текст

Исключения

cipher_error	Если тексты не соответствуют требованиям
------------------------------	--

Требования:

- Длина шифртекста должна совпадать с длиной открытого текста
- Должен содержать только английские буквы
- Не должен быть пустым

Аргументы

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

Возвращает

Валидный зашифрованный текст

Исключения

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

Основная проверка - соответствие длины шифртекста длине открытого текста. Символьная валидация выполняется в вызывающем методе.

4.1.3.3 getKey()

```
int Cipher::getKey (
    int key,
    const string & Text ) [inline], [private]
```

Проверяет валидность ключа шифрования

Аргументы

key	Предлагаемый ключ (количество столбцов)
Text	Текст для шифрования

Возвращает

Валидный ключ

Исключения

<code>cipher_error</code>	Если ключ невалиден
---------------------------	---------------------

Требования к ключу:

- Должен быть в диапазоне $[2, \text{длина_текста}]$
- Не может быть меньше 2
- Не может быть больше длины текста

Аргументы

key	Предлагаемый ключ (количество столбцов)
Text	Текст для шифрования

Возвращает

Валидный ключ

Исключения

<code>cipher_error</code>	Если ключ невалиден
---------------------------	---------------------

Ключ должен удовлетворять условиям:

- $\text{key} \geq 2$ (минимум 2 столбца)
- $\text{key} \leq \text{Text.length}()$ (не больше длины текста)

Это обеспечивает корректное формирование таблицы для шифрования.

4.1.3.4 `getValidOpenText()`

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

Проверяет и нормализует открытый текст

Аргументы

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

Возвращает

Валидный открытый текст без пробелов

Исключения

<code>cipher_error</code>	Если текст пуст или содержит недопустимые символы
---------------------------	---

Особенности обработки:

- Удаляет все пробелы
- Проверяет наличие только английских букв
- Сохраняет оригинальный регистр символов

Аргументы

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

Возвращает

Валидный открытый текст без пробелов

Исключения

<code>cipher_error</code>	Если текст пуст или содержит недопустимые символы
---------------------------	---

Выполняет:

- Проверку на пустоту
- Удаление пробелов
- Проверку на наличие только английских букв
- Сохранение оригинального регистра

4.1.3.5 transcript()

```
string Cipher::transcript (  
    string & text,  
    string & open_text )
```

Расшифровывает зашифрованный текст

Аргументы

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

Возвращает

Расшифрованная строка

Исключения

cipher_error	Если тексты невалидны или длины не совпадают
------------------------------	--

Алгоритм расшифрования обратен алгоритму шифрования:

1. Валидация шифртекста и открытого текста
2. Создание таблицы $\text{key} \times (\text{n}/\text{key})$ столбцов
3. Запись шифртекста в таблицу по столбцам справа налево
4. Считывание текста по строкам слева направо
5. Удаление символов-заполнителей '0'

Аргументы

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

Возвращает

Расшифрованная строка

Исключения

cipher_error	Если тексты невалидны или длины не совпадают
------------------------------	--

Выполняет обратное преобразование:

1. Валидация обоих текстов
2. Создание динамической таблицы
3. Запись шифртекста в таблицу по столбцам справа налево
4. Считывание текста из таблицы по строкам слева направо
5. Освобождение памяти таблицы
6. Удаление символов-заполнителей '0'

Заметки

Длина шифртекста должна совпадать с длиной открытого текста
Пустые ячейки заполняются символом '0'

См. также

[getValidCipherText](#)

Объявления и описания членов классов находятся в файлах:

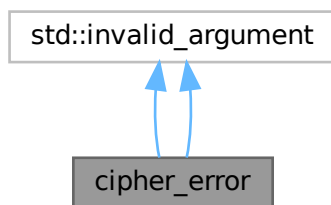
- [TableRouteCipher.h](#)
- [TableRouteCipher.cpp](#)

4.2 Класс cipher_error

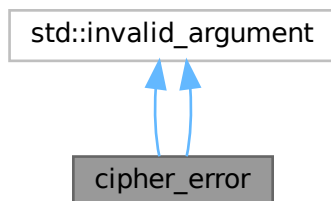
Класс исключений для ошибок шифрования

```
#include <TableRouteCipher.h>
```

Граф наследования: cipher_error:



Граф связей класса cipher_error:



Открытые члены

- [cipher_error](#) (const string &what_arg)
Конструктор с строковым параметром
- [cipher_error](#) (const char *what_arg)
Конструктор с C-строкой

4.2.1 Подробное описание

Класс исключений для ошибок шифрования

Используется для обработки ошибок, связанных с невалидными ключами, текстами и другими проблемами при работе с шифром маршрутной перестановки.

4.2.2 Конструктор(ы)

4.2.2.1 cipher_error() [1/2]

```

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

```

Конструктор с строковым параметром

Аргументы

what_arg	Сообщение об ошибке в виде std::string
----------	--

4.2.2.2 cipher_error() [2/2]

```

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

```

Конструктор с C-строкой

Аргументы

what_arg	Сообщение об ошибке в виде C-строки
----------	-------------------------------------

Объявления и описания членов класса находятся в файле:

- [TableRouteCipher.h](#)

Глава 5

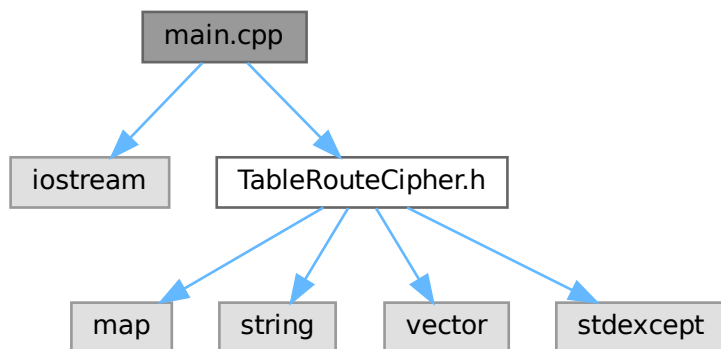
Файлы

5.1 Файл main.cpp

Демонстрационная программа для тестирования шифра маршрутной перестановки

```
#include <iostream>
#include "TableRouteCipher.h"
```

Граф включаемых заголовочных файлов для main.cpp:



Функции

- void `check` (string Text, int key)
Тестирует работу шифра с заданными параметрами
- int `main` ()
Главная функция программы

5.1.1 Подробное описание

Демонстрационная программа для тестирования шифра маршрутной перестановки

Содержит функцию для тестирования работы класса [Cipher](#) с различными входными данными и обработкой исключений.

Автор

Жбанчиков

Дата

2025

Версия

1.0

5.1.2 Функции

5.1.2.1 `check()`

```
void check (
    string Text,
    int key )
```

Тестирует работу шифра с заданными параметрами

Аргументы

Text	Открытый текст для шифрования
key	Ключ шифрования (количество столбцов)

Выполняет полный цикл шифрования и расшифрования:

1. Создает объект шифратора с заданным ключом
2. Шифрует текст методом маршрутной перестановки
3. Расшифровывает полученный шифртекст
4. Выводит исходный текст, зашифрованный и расшифрованный тексты
5. Обрабатывает возможные исключения

Заметки

Функция демонстрирует симметричность шифрования: `decrypt(encrypt(text)) == text`

Пример вывода:

Ключ = 4

Исходный текст: ArbekovoArlekino

Зашифрованный текст: [зашифрованный_текст]

Расшифрованный текст: ArbekovoArlekino

5.1.2.2 main()

```
int main ( )
```

Главная функция программы

Возвращает

Код возврата: 0 при успешном выполнении

Выполняет тестирование шифра с конкретными входными данными:

- Текст: "ArbekovoArlekino"
- Ключ: 4

Программа демонстрирует:

- Корректность работы алгоритма шифрования
- Симметричность операций шифрования/расшифрования
- Обработку исключительных ситуаций
- Работу с английским алфавитом в смешанном регистре

Заметки

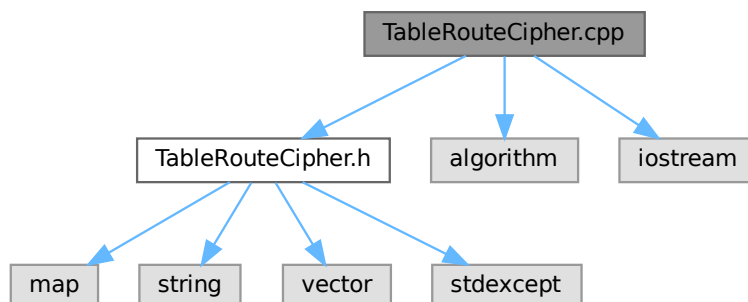
Для дополнительного тестирования можно добавить вызовы функции `check()` с другими параметрами.

5.2 Файл TableRouteCipher.cpp

Реализация класса шифрования методом маршрутной перестановки

```
#include "TableRouteCipher.h"  
#include <algorithm>  
#include <iostream>
```

Граф включаемых заголовочных файлов для TableRouteCipher.cpp:



5.2.1 Подробное описание

Реализация класса шифрования методом маршрутной перестановки

Содержит реализацию всех методов класса [Cipher](#). Включает алгоритмы шифрования/расшифрования, работу с таблицами и валидацию входных данных.

Автор

Жбанчиков

Дата

2025

Версия

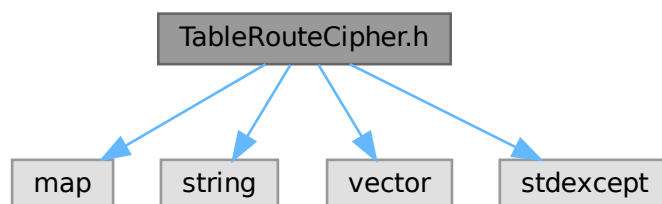
1.0

5.3 Файл TableRouteCipher.h

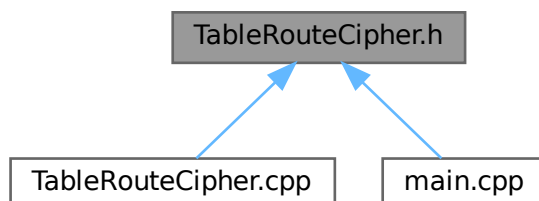
Заголовочный файл для класса шифрования методом маршрутной перестановки

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

Граф включаемых заголовочных файлов для TableRouteCipher.h:



Граф файлов, в которые включается этот файл:



Классы

- class `cipher_error`
Класс исключений для ошибок шифрования
- class `Cipher`
Класс для шифрования методом маршрутной перестановки

5.3.1 Подробное описание

Заголовочный файл для класса шифрования методом маршрутной перестановки

Реализует табличное шифрование с маршрутным считыванием. Текст записывается в таблицу по строкам и считывается по столбцам в обратном порядке для шифрования.

Автор

Анохин

Дата

2025

Версия

1.0

5.4 TableRouteCipher.h

[См. документацию.](#)

```

00001
00012 #pragma once
00013 #include <map>
00014 #include <string>
00015 #include <vector>
00016 #include <stdexcept>
00017 using namespace std;
00018
00027 class cipher_error : public invalid_argument {
00028 public:
00033     explicit cipher_error(const string& what_arg)
00034         : invalid_argument(what_arg) {}
00035
00040     explicit cipher_error(const char* what_arg)
00041         : invalid_argument(what_arg) {}
00042 };
00043
00063 class Cipher {
00064 private:
00065     int key;
00066
00079     inline int getValidKey(int key, const string& Text);
00080
00092     inline string getValidOpenText(string& s);
00093
00106     inline string getValidCipherText(string& s, string& open_text);
00107
00108 public:
00113     Cipher() = delete;
00114
00126     Cipher(int skey, string text);
00127
00146     string encryption(string& text);
00147
00162     string transcript(string& text, string& open_text);
00163 };

```


Предметный указатель

- check
 - main.cpp, [18](#)
- Cipher, [7](#)
 - Cipher, [8](#)
 - encryption, [9](#)
 - getValidCipherText, [10](#)
 - getValidKey, [11](#)
 - getValidOpenText, [12](#)
 - transcript, [13](#)
- cipher_error, [15](#)
 - cipher_error, [16](#)
- encryption
 - Cipher, [9](#)
- getValidCipherText
 - Cipher, [10](#)
- getValidKey
 - Cipher, [11](#)
- getValidOpenText
 - Cipher, [12](#)
- main
 - main.cpp, [18](#)
- main.cpp, [17](#)
 - check, [18](#)
 - main, [18](#)
- TableRouteCipher.cpp, [19](#)
- TableRouteCipher.h, [20](#)
- transcript
 - Cipher, [13](#)