

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационная безопасность систем и технологий»

Отчет  
о лабораторной работе №4  
на тему «Документирование программы»

Дисциплина: Методы и средства  
программирования

Группа: 22ПИ2

Выполнил: Никитина М. А.

Количество баллов:

Дата сдачи:

Принял: Н. А. Сидоров

2023

1 Цель работы: освоить документирование программы на языке Си++ с использованием программы doxygen.

2 Задание на лабораторную работу

2.1 Сформировать блоки документирования для ранее разработанных модулей.

2.2 Сформировать документацию в форматах HTML и PDF.

3 Порядок выполнения работы

3.1 Были сформированы блоки документирования для ранее разработанных модулей. Были изменены параметры:

а) PROJECT\_NAME = "Cipher";

б) PROJECT\_NUMBER = 1;

в) OUTPUT\_LANGUAGE = Russian;

г) EXTRACT\_PRIVATE = YES.

3.2 Была сформирована документация в форматах HTML и PDF. PDF формат документации был создан с помощью функции make. Страница html представлена на рисунке 1. Код программы с комментариями для документации представлен в приложении А. Каталог, содержащий index.html, и refman.pdf, содержащий документацию, представлены на github.

← → ↻

file:///home/stud/C++Projects/UT33/html/main\_8cpp.html

# Cipher 1

Титульная страница

Классы ▾

Файлы ▾

## Файл main.cpp

Главный модуль программы [Подробнее...](#)

```
#include <iostream>
#include <UnitTest++/UnitTest++.h>
#include "cipher.h"
```

Граф включаемых заголовочных файлов для main.cpp:

iostre

## Функции

string	<b>check</b>	(int key, const string &msg)
int	<b>runTests</b>	()
	<b>SUITE</b>	(KeyTest)

Рисунок 1 - Документация

## 4 Выводы

С помощью данной лабораторной работы был освоено документирование программы на языке Си++ с использованием программы doxygen. Полный результат работы можно посмотреть по ссылке: <https://github.com/Popitka994/MiSP/tree/main/Lb4>.

## Приложение А

Код модуля main.cpp:

```
/** @file
 * @brief Главный модуль программы
 * @details Есть функция check и модульные тесты KeyTest,
 * TextTest.
 */
#include <iostream>
#include <UnitTest++/UnitTest++.h>
#include "cipher.h"
using namespace std;

/** @file
 * @brief Функция check
 * @details Функция была создана в ходе 2-ой лабораторной
 * работы.
 */
string check(int key, const string& msg)
{
    try {
        cout << "Ключ: " << key << endl;
        if (key <= 1) throw invalid_argument("Ключ должен
        быть целым числом больше единицы!");
        if (msg.empty()) throw invalid_argument("Исходный
        текст не может быть пустым!");
        Cipher cipher(key);
        cout << "Исходный текст: " << msg << endl;
        string encrypted = cipher.encrypt(msg);
```

```

        cout << "Зашифрованный текст: " << encrypted <<
endl;
        string decrypted = cipher.decrypt(encrypted);
        cout << "Расшифрованный текст: " << decrypted <<
endl;
        return encrypted;
    } catch (const exception& e) {
        cerr << "Error: " << e.what() << endl;
        return "";
    }
}

/** @file
 * @brief Модульные тесты
 * @details KeyTest проверяют ключ на длины. TextTest
проверяет исходный текст на регистр и наличие символов.
 */
int runTests()
{
    return UnitTest::RunAllTests();
}

//Проверка ключа
SUITE(KeyTest)
{
    TEST(ValidKey) {
        CHECK_EQUAL("rvtpie",    check(2,
"privet")); //Верный ключ
    }
}

```

```

    TEST(LongerKeyThanMessage) {
        CHECK_EQUAL("hello", check(10, "hello")); //Ключ
        больше, чем длина
    }

    TEST(KeyLessOne) {
        CHECK_THROW(check(1, "hello"), invalid_argument);
        //Ключ меньше 1
    }
}

//Проверка исходного текста
SUITE(TextTest) {
    TEST(UppercaseLetters) {
        CHECK_EQUAL("RVTPIE", check(2,
        "PRIVET")); //Прописные буквы
    }
    TEST(LowercaseLetters) {
        CHECK_EQUAL("rvtpie", check(2,
        "privet")); //Строчные
    }
    TEST(NonAlphabeticCharacters) {
        CHECK_THROW(check(2, "pri12!"),
        invalid_argument); //Есть неалфавитные символы
    }
    TEST(EmptyText) {
        CHECK_THROW(check(2, ""),
        invalid_argument); //Пустая строка
    }
}

```

```

    }
    TEST(NonAlphabeticLetters) {
        CHECK_THROW(check(2, "123!@#"),
invalid_argument); //Нет букв
    }
}

```

```

int main(int argc, char **argv)
{
    runTests();
    return 0;
}

```

Код модуля cipher.cpp:

```

/** @file
 * @brief Модуль cipher.cpp
 * @details Реализация методов класса Cipher для
шифрования и дешифрования методом маршрутной
перестановки.
 */
#include "cipher.h"
#include <string>
using namespace std;

Cipher::Cipher(int k) : key(k) {}

/** @file
 * @brief Шифрование
 * @param msg Исходный текст для шифрования.
 * @return Зашифрованный текст.

```

```

*/
string Cipher::encrypt(const string& msg) {
    int kolvo_strok = (msg.length() + key - 1) / key;
    size_t index = 0;
    char table[kolvo_strok][key];

    for (int i = 0; i < kolvo_strok; i++) {
        for (int j = 0; j < key; j++) {
            if (index < msg.length()) {
                table[i][j] = msg[index];
                index++;
            } else {
                table[i][j] = ' ';
            }
        }
    }

    string encrypted;
    for (int j = key - 1; j < key && j >= 0; j = j - 1) {
        for (int i = 0; i < kolvo_strok; i++) {
            encrypted += table[i][j];
        }
    }
    return encrypted;
}

/** @file
 * @brief Дешифрование
 * @param encrypted Зашифрованный текст.

```



```

* @return Расшифрованный текст.
*/
string Cipher::decrypt(const string& encrypted) {
    int kolvo_strok = encrypted.length() / key;
    size_t index = 0;
    char table[kolvo_strok][key];

    for (int j = key - 1; j < key && j >= 0; j = j - 1) {
        for (int i = 0; i < kolvo_strok; i++) {
            table[i][j] = encrypted[index];
            index++;
        }
    }

    string msg;
    for (int i = 0; i < kolvo_strok; i++) {
        for (int j = 0; j < key; j++) {
            msg += table[i][j];
        }
    }
    return msg;
}

```

Код модуля cipher.h:

```

/** @file
* @author Никитина М. А.
* @version 1.0
* @date 15.12.23
* @copyright ИБСТ ПГУ

```

```

*  @brief Заголовочный файл для модуля по методу
маршрутной перестановки
*/
#ifndef CIPHER_H
#define CIPHER_H
#include <string>
using namespace std;

/**
*  @brief Конструктор класса Cipher
*  #param k Ключ для шифрования.
*/
class Cipher {
private:
    int key;
public:
    Cipher(int k);
    string encrypt(const string& msg);
    string decrypt(const string& zashifrovan);
};
#endif

```