

Министерство образования и науки РФ
ФГБОУ ВО ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра «Информационная безопасность систем и технологий»

ОТЧЕТ
о лабораторной работе №3
МОДУЛЬНОЕ ТЕСТИРОВАНИЕ

Дисциплина: Технологии и методы
программирования

Группа: 18ПИ1

Выполнил: Новиков Д.О.

Количество баллов:

Дата сдачи:

Принял: к.т.н., доцент Лупанов М. Ю.

Пенза 2019

1 Цель работы

1.1 Освоить процесс модульного тестирования разрабатываемых программ .

2 Задание к лабораторной работе

2.1 Адаптировать приведенные тестовые сценарии к модулю шифрования русскоязычных сообщений методом Гронсвельда, разработанному при выполнении предыдущих работ и выполнить модульное тестирование.

2.2 Разработать тестовые сценарии для модуля шифрования методом маршрутной перестановки, разработанного при выполнении предыдущих работ.

2.3 Разработать модульные тесты и провести тестирование модуля шифрования методом маршрутной перестановки

3 Результаты работы

3.1 Имея готовые тестовые сценарии, приведенные в содержании лабораторной работы 3, создаем модульный тест. Для этого в CodeLite создаем проект UnitTest++. После создания проекта добавляем в него тестируемый модуль. Это файлы modAlphaCipher.cpp и modAlphaCipher.h. Для доступа к модулю из главной программы подключаем заголовочный файл modAlphaCipher.h. Далее для каждого тестового сценария создаем с помощью макроса SUITE набор тестов. Тесты и полный текст файла main.cpp представлен в Приложении А. Результат модульного тестирования представлен на Рисунке 1.

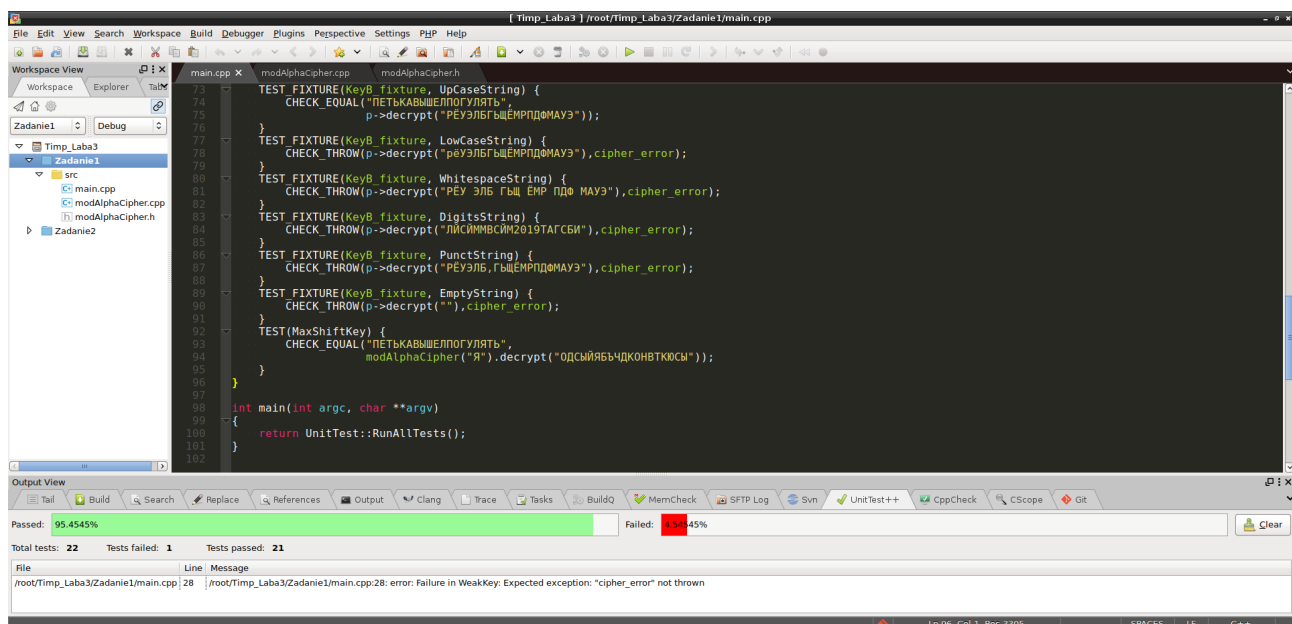


Рисунок 1 - Результат модульного тестирования 1

Из Рисунка 1 видно, что один тест не прошел. В диагностике указываются строки для «заваленных» тестов. В данном случае, это TEST(WeakKey) — тест на вырожденный ключ. Для обнаруженного дефекта добавляем в конец конструктора код представленный на Рисунке 2.

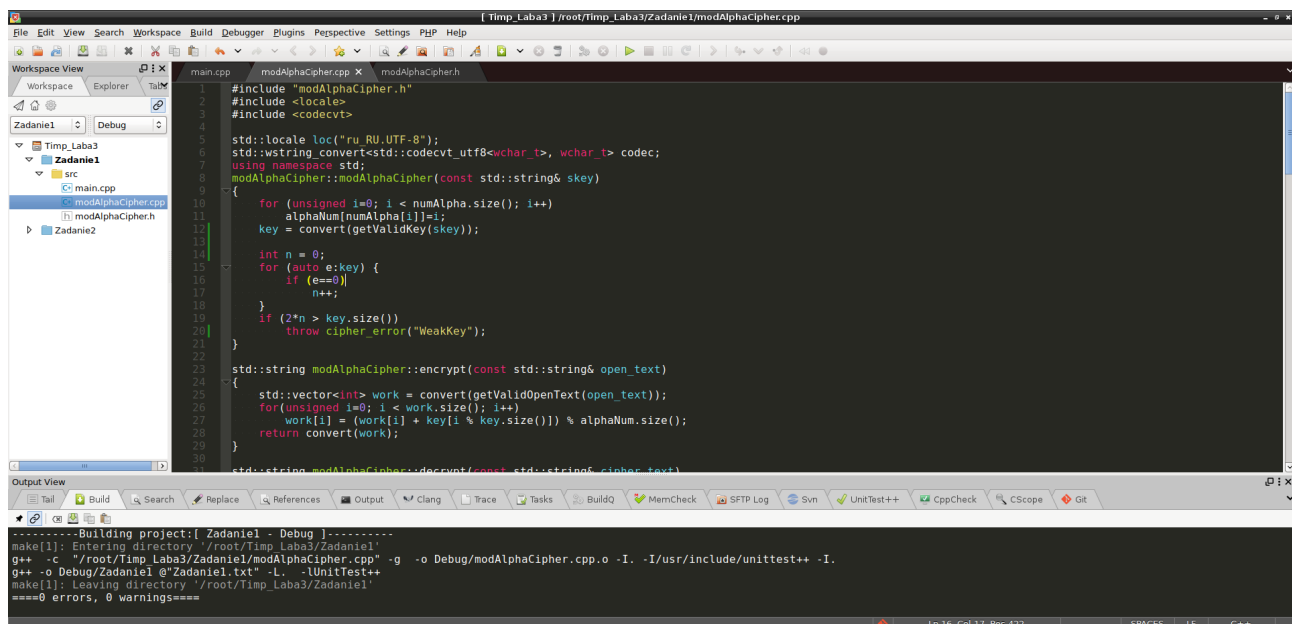


Рисунок 2 - Исправленный конструктор

В этом случае исключение будет возбуждаться, если больше половины символов ключа не преобразуют открытый текст. После внедрения такой

«заплатки» в код программы повторное тестирование дефектов не выявило. Не выявленные дефекты представлены на Рисунке 3.

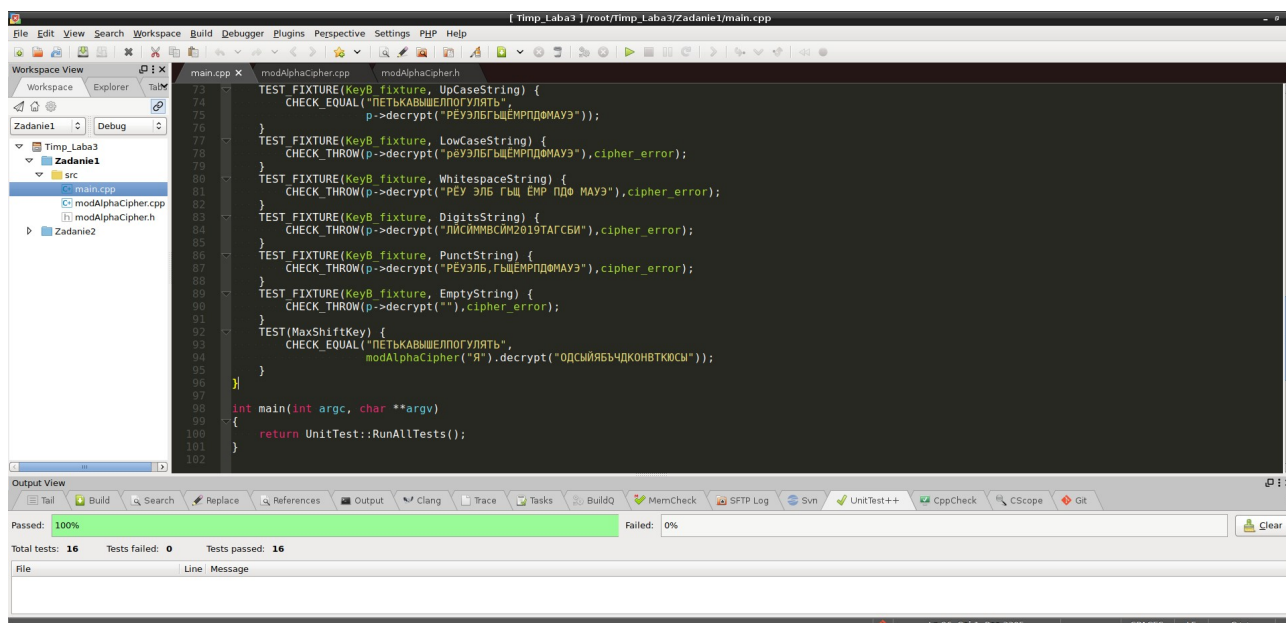


Рисунок 3 - Результат модульного тестирования 2

3.2 Создадим тестовый сценарий для тестирования модуля, приведенного в предыдущей работе. Для этого проанализируем код модуля. В открытой секции класса находятся конструктор с параметром и методы `encrypt` и `decrypt`. Соответственно, нам необходимо три тестовых сценария для их проверки. Тестовый сценарий для конструктора представлен на Таблице 1. Тестовый сценарий для метода `encrypt` представлен на Таблице 2. Тестовый сценарий метода `decrypt` представлен на Таблице 1.

Таблица 1 - Тестовый сценарий для конструктора

№	Тест	Параметр конструктора	Параметр метода encrypt	Ожидаемый результат (конструктор)	Ожидаемый результат (encrypt)
1.1	Верный ключ	7	ПЕТЬКАВЫШЕЛ ПОГУЛЯТЬ	-	ВГАОКПЬЛТТЕЯ ЕШЛПЫУ
1.2	Ключ длиннее сообщения	100	ПЕТЬКАВЫШЕЛ ПОГУЛЯТЬ	-	БТЯЛУГОПЛЕШЫ ВАКЬТЕП
1.3	В ключе отрицательное число	-245	-	исключение	-
1.4	В ключе ноль	0	-	исключение	-

Таблица 2 - Тестовый сценарий для метода encrypt

№	Тест	Ключ	Параметр метода encrypt	Ожидаемый результат (encrypt)
2.1	Строка из прописных	5	ПЕТЬКАВЫШЕЛПОГУЛЯТЬ	КЕУЫШГЪТЫОТЕВПЯПА ЛЛ
2.2	Строка из строчных	5	петькавышелпогулять	КЕУЫШГЪТЫОТЕВПЯПА ЛЛ
2.3	Строка с пробелами и знаками препинания	5	Петька вышел погулять!!!	КЕУЫШГЪТЫОТЕВПЯПА ЛЛ
2.4	Строка с цифрами	5	Кирилл брился в 2019 раз	ЛЛАИИРРРВИБЯКЛСЗ
2.5	Пустой текст	5	Пустая строка	исключение
2.6	Нет букв	5	1234+8765=9999	исключение
2.7	Максимальный сдвиг	23852	ПЕТЬКАВЫШЕЛПОГУЛЯТЬ	БТЯЛУГОПЛЕШЫВАКЬТЕ П

Таблица 3 - Тестовый сценарий для метода decrypt

№	Тест	Ключ	Параметр метода decrypt	Ожидаемый результат (decrypt)
3.1	Есть строчные	5	кеУЫШГЪСЫОТАВПЯВАЛЛ	исключение
3.2	Есть пробелы	5	КЕУ ЫШГ ЪТЫ ОТЕ ВПЯ ПАЛЛ	исключение
3.3	Есть цифры	5	ЛЛАИИРРРВИБ2019ЯКЛСЗ	исключение
3.4	Есть знаки препинания	5	КЕУЫШГ,ЪТЫОТЕВПЯПАЛЛ	исключение
3.5	Пустой текст	5	Пустая строка	исключение

3.3 Имея готовые тестовые сценарии, приведенные в пункте 3.2, создаем модульный тест. Для этого в CodeLite создаем проект UnitTest++. После создания проекта добавляем в него тестируемый модуль. Это файлы `swarcipher.cpp` и `swarcipher.h`. Для доступа к модулю из главной программы подключаем заголовочный файл `swarcipher.h`. Далее для каждого тестового сценария создаем с помощью макроса `SUITE` набор тестов. Тесты и полный текст файла `main.cpp` представлен в Приложении Б. Результат модульного тестирования представлен на Рисунке 4.

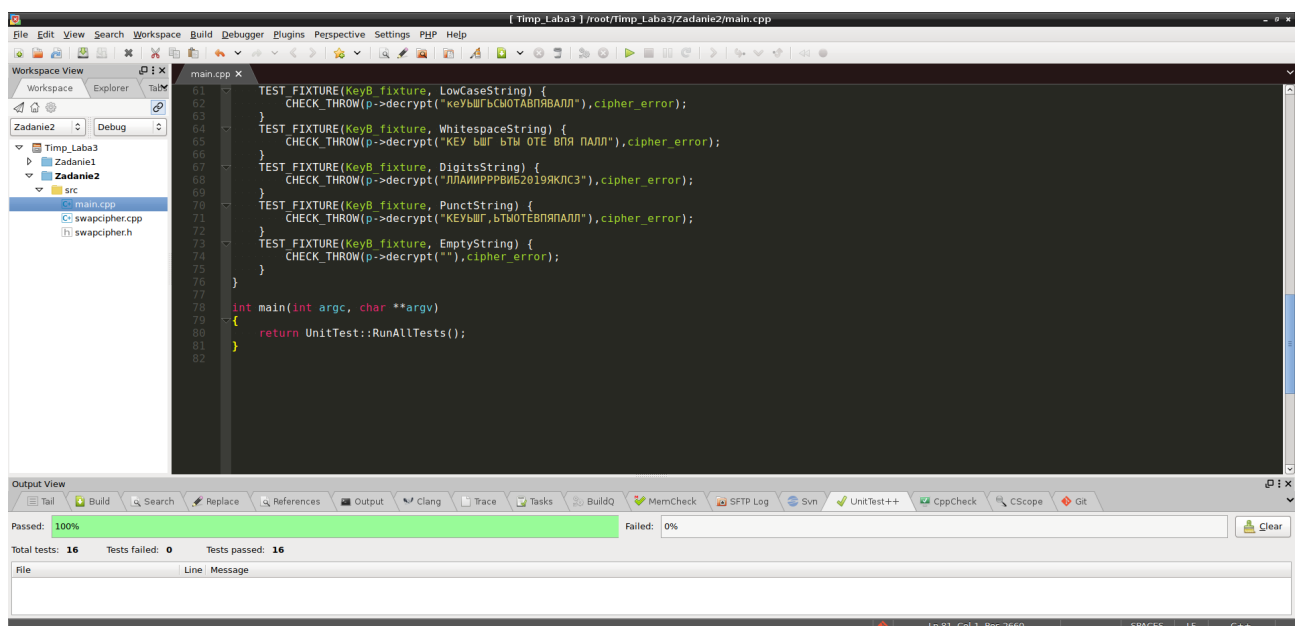


Рисунок 4 - Результат модульного тестирования табличной маршрутной перестановки

4 Вывод

В результате выполнения работы был освоен процесс модульного тестирования языка Си++, а также были разработаны тестовые сценарии для многомодульного проекта маршрутной перестановки, и получены практические навыки в создании модульных тестов.

Приложение А

Текст файла main.cpp

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

SUITE(KeyTest)
{
    TEST(ValidKey) {
        CHECK_EQUAL("БЦДБЦ", modAlphaCipher("БЦД").encrypt("AAAAA"))
    };

    TEST(LongKey) {
        CHECK_EQUAL("БЦДЕФ", modAlphaCipher("БЦДЕФГШИЖК").encrypt("AAAAA"));
    };

    TEST(LowCaseKey) {
        CHECK_EQUAL("БЦДБЦ", modAlphaCipher("бцд").encrypt("AAAAA"))
    };

    TEST(DigitsInKey) {
        CHECK_THROW(modAlphaCipher cp("Б1"), cipher_error);
    };

    TEST(PunctuationInKey) {
        CHECK_THROW(modAlphaCipher cp("Б,С"), cipher_error);
    };

    TEST(WhitespaceInKey) {
        CHECK_THROW(modAlphaCipher cp("Б С"), cipher_error);
    };

    TEST(EmptyKey) {
        CHECK_THROW(modAlphaCipher cp(""), cipher_error);
    };

    TEST(WeakKey) {
        CHECK_THROW(modAlphaCipher cp("AAA"), cipher_error);
    };
}
```

```

    }
}

struct KeyB_fixture {
    modAlphaCipher * p;
    KeyB_fixture() {
        p = new modAlphaCipher("B");
    }
    ~KeyB_fixture() {
        delete p;
    }
};

SUITE(EncryptTest)
{
    TEST_FIXTURE(KeyB_fixture, UpCaseString) {
        CHECK_EQUAL("РЁУЭЛБГЫЩЁМРПДФМАУЭ",
                    p->encrypt("ПЕТЬКАВЫШЕЛПОГУЛЯТЬ"));
    }
    TEST_FIXTURE(KeyB_fixture, LowCaseString) {
        CHECK_EQUAL("РЁУЭЛБГЫЩЁМРПДФМАУЭ",
                    p->encrypt("петькавышелпогулять"));
    }
    TEST_FIXTURE(KeyB_fixture, StringWithWhitspaceAndPunct) {
        CHECK_EQUAL("РЁУЭЛБГЫЩЁМРПДФМАУЭ",
                    p->encrypt("Петька вышел погулять!!!"));
    }
    TEST_FIXTURE(KeyB_fixture, StringWithNumbers) {
        CHECK_EQUAL("ЛЙСЙММВСЙМТАГСБИ", p->encrypt("Кирилл брился
в 2019 раз"));
    }
    TEST_FIXTURE(KeyB_fixture, EmptyString) {
        CHECK_THROW(p->encrypt(""), cipher_error);
    }
}

```



```

    }
    TEST_FIXTURE(KeyB_fixture, NoAlphaString) {
        CHECK_THROW(p->encrypt("1234+8765=9999"), cipher_error);
    }
    TEST(MaxShiftKey) {
        CHECK_EQUAL("ОДСЫЙЯБЪЧДКОНВТКЮСЫ",
                    modAlphaCipher("Я").encrypt("ПЕТЬКАВЫШЕЛПОГУЛЯ
ТЬ")));
    }
}

SUITE(DecryptText)
{
    TEST_FIXTURE(KeyB_fixture, UpCaseString) {
        CHECK_EQUAL("ПЕТЬКАВЫШЕЛПОГУЛЯТЬ",
                    p->decrypt("РЁУЭЛБГЬЩЁМРПДФМАУЭ"));
    }
    TEST_FIXTURE(KeyB_fixture, LowCaseString) {
        CHECK_THROW(p->decrypt("рёуэлбгьщёмрпдфмауэ"), cipher_error);
    }
    TEST_FIXTURE(KeyB_fixture, WhitespaceString) {
        CHECK_THROW(p->decrypt("РЁУ  ЭЛБ  ГЬЩ  ЁМР  ПДФ
МАУЭ"), cipher_error);
    }
    TEST_FIXTURE(KeyB_fixture, DigitsString) {
        CHECK_THROW(p->decrypt("ЛЙСЙММВСЙМ2019ТАГСВИ"), cipher_error);
    }
    TEST_FIXTURE(KeyB_fixture, PunctString) {
        CHECK_THROW(p->decrypt("РЁУЭЛБ,ГЬЩЁМРПДФМАУЭ"), cipher_error);
    }
}

```

```

TEST_FIXTURE(KeyB_fixture, EmptyString) {
    CHECK_THROW(p->decrypt(""), cipher_error);
}

TEST(MaxShiftKey) {
    CHECK_EQUAL("ПЕТЬКАВЫШЕЛПОГУЛЯТЬ",
                modAlphaCipher("Я").decrypt("ОДСЫЙЯБЪЧДКОНВТКЮ
СЫ"));
}

int main(int argc, char **argv)
{
    return UnitTest::RunAllTests();
}

```

Приложение Б

Текст файла main.cpp табличной маршрутной перестановки

```
#include <unittest++/UnitTest++.h>
#include "swapcipher.h"

SUITE(KeyTest)
{
    TEST(ValidKey) {
        CHECK_EQUAL("ВГАОКПЬЛТТЕЯЕШЛПЫУ", SwapCipher(7).encrypt("П
ЕТЬКАВЫШЕЛПОГУЛЯТЬ"));
    }
    TEST(LongKey) {
        CHECK_EQUAL("ЪТЯЛУГОПЛЕШЫВАКЪТЕП", SwapCipher(100).encrypt(
"ПЕТЬКАВЫШЕЛПОГУЛЯТЬ"));
    }
    TEST(MinusInKey) {
        CHECK_THROW(SwapCipher cp(-245), cipher_error);
    }
    TEST(ZeroInKey) {
        CHECK_THROW(SwapCipher cp(0), cipher_error);
    }
}

struct KeyB_fixture {
    SwapCipher * p;
    KeyB_fixture() {
        p = new SwapCipher(5);
    }
    ~KeyB_fixture() {
        delete p;
    }
};
```

```

SUITE(EncryptTest)
{
    TEST_FIXTURE(KeyB_fixture, UpCaseString) {
        CHECK_EQUAL("КЕУЫШГЪТЫОТЕВПЯПАЛЛ",
                    p->encrypt("ПЕТЬКАВЫШЕЛПОГУЛЯТЬ"));
    }
    TEST_FIXTURE(KeyB_fixture, LowCaseString) {
        CHECK_EQUAL("КЕУЫШГЪТЫОТЕВПЯПАЛЛ",
                    p->encrypt("петькавышелпогулять"));
    }
    TEST_FIXTURE(KeyB_fixture, StringWithWhitspaceAndPunct) {
        CHECK_EQUAL("КЕУЫШГЪТЫОТЕВПЯПАЛЛ",
                    p->encrypt("Петька вышел погулять!!!"));
    }
    TEST_FIXTURE(KeyB_fixture, StringWithNumbers) {
        CHECK_EQUAL("ЛЛАИИРРРВИВЯКЛСЗ", p->encrypt("Кирилл брился
в 2019 раз"));
    }
    TEST_FIXTURE(KeyB_fixture, EmptyString) {
        CHECK_THROW(p->encrypt(""), cipher_error);
    }
    TEST_FIXTURE(KeyB_fixture, NoAlphaString) {
        CHECK_THROW(p->encrypt("1234+8765=9999"), cipher_error);
    }
    TEST(MaxShiftKey) {
        CHECK_EQUAL("ЪТЯЛУГОПЛЕШЫВАКЪТЕП",
                    SwapCipher(23852).encrypt("ПЕТЬКАВЫШЕЛПОГУЛЯТЬ
"));
    }
}

SUITE(DecryptText)
{

```

```

    TEST_FIXTURE(KeyB_fixture, LowCaseString) {
        CHECK_THROW(p->decrypt("кеуышгъсыотавпявалл"), cipher_error);
    }
    TEST_FIXTURE(KeyB_fixture, WhitespaceString) {
        CHECK_THROW(p->decrypt("КЕУ  ЫШГ  ЫТЫ  ОТЕ  ВПЯ
ПАЛЛ"), cipher_error);
    }
    TEST_FIXTURE(KeyB_fixture, DigitsString) {
        CHECK_THROW(p->decrypt("ЛЛАИИРРРВИБ2019ЯКЛСЗ"), cipher_error);
    }
    TEST_FIXTURE(KeyB_fixture, PunctString) {
        CHECK_THROW(p->decrypt("КЕУЫШГ,ЫТЫОТЕВПЯПАЛЛ"), cipher_error);
    }
    TEST_FIXTURE(KeyB_fixture, EmptyString) {
        CHECK_THROW(p->decrypt(""), cipher_error);
    }
}

int main(int argc, char **argv)
{
    return UnitTest::RunAllTests();
}

```