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

ОТЧЁТ
о лабораторной работе №3
Модульное тестирование.

Дисциплина: Технологии и методы программирования
Группа:18ПИ1
Выполнил: Асаян А.В.
Количество баллов:
Дата сдачи:
Проверил: к.т.н., доцент Лупанов М.Ю.

Пенза, 2019

1 Цель работы

1.1 Освоить процесс модульного тестирования разрабатываемых программ.

2 Задания к практической 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 Результат выполнения работы

3.1 Тестовые сценарии, приведённые в качестве примера в методических указаниях были адаптированы для тестов шифра Гронсвельда на русском языке. Для проведения тестирования были изменены методы encrypt, decrypt и convert, так чтобы они объект класса string, а не wstring. Так же в программу было добавлено исключение, которое возбуждается, если больше половины символов ключа не изменяют исходный текст.

Таблица 1.-Тестовый сценарий для конструктора.

№	Тест	Параметры конструктора	Параметры метода encrypt	Ожидаемый результат(конструктор)	Ожидаемый результат(encrypt)
1.1	Верный ключ	БВГ	ААААА	-	БВГБВ
1.2	Ключ длиннее сообщения	БВГДЕЖЗИК	ААААА		БВГДЕ
1.3	В ключе строчные буквы	бвг	ААААА	-	БВГБВ
1.4	В ключе цифры	В1	-	исключение	-
1.5	В ключе знаки препинания	В,С	-	исключение	-
1.6	В ключе пробелы	В С	-	исключение	-

1.7	Пустой ключ		-	исключение	-
1.8	Вырожденный ключ	AAA	ПРИВЕТ	исключение	

Таблица 2.-Тестовый сценарий для метода encrypt.

№	Тест	Ключ	Параметры метода encrypt	Ожидаемый результат encrypt
2.1	Строка из прописных букв	Б	СЪЕШЬЕЩЁЭТИМЯГКИХФРАНЦУЗСКИХБУЛОКДАВЫПЕЙЧАЮЖАК	ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМПЛЕБГЬРЁКШБЯЗБЛ
2.2	Строка из строчных букв	Б	съешьещёэтимягкихфранцузскихбулокдавыпейчаюжак	ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМПЛЕБГЬРЁКШБЯЗБЛ
2.3	Строк с пробелами и знаками препинания	Б	Съешь ещё эти мягких французских булок,да выпей чаю, жак	ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМПЛЕБГЬРЁКШБЯЗБЛ
2.4	Строка с цифра	Б	1Съешь 2ещё 3эти 4мягких 5французских 6булок, 7да 8выпей 9чаю, жак	ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМПЛЕБГЬРЁКШБЯЗБЛ
2.5	Пустая строка	Б		исключение
2.6	Строка без букв	Б	214+567	исключение
2.7	Максимальный сдвиг	Я	СЪЕШЬЕЩЁЭТИМЯГКИХФРАНЦУЗСКИХБУЛОКДАВЫПЕЙЧАЮЖАК	РЩДЧЫДШЕЬСЗЛЮВЙЗФУПЯМХТЖРЙЗФАТКНЙГЯБЪОДИЦЯЭЁЯЙ

Таблица 3. - Тестовый сценарий для метода decrypt.

№	Тест	Ключ	Параметры метода encrypt	Ожидаемый результат encrypt
3.1	Строка из прописных букв	Б	ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМПЛЕБГЬРЁКШБЯЗБЛ	СЪЕШЬЕЩЁЭТИМЯГКИХФРАНЦУЗСКИХБУЛОКДАВЫПЕЙЧАЮЖАК
3.2	Строка со строчными буквами	Б	тыЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМПЛЕБГЬРЁКШБЯЗБЛ	исключение
3.3	Строк с пробелами	Б	ТЫ ЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВ	исключение

			ФМ ПЛЕБГЪРЁКШБЯЗБЛ	
3.4	Строка с цифра	Б	ТЫЁЩЭ2ЁЪЖЮУЙНАДЛЙЦ1ХСБОЧФИТЛЙЦВФМПЛЕБГЬ4РЁКШБЯЗБ5Л	исключение
3.5	Пустая строка	Б		исключение
3.6	Строка со знаками препинания	Б	ТЫЁЩ,ЭЁЪЖЮУЙНАДЛЙЦ,ХСБОЧФИТЛЙЦВФМПЛЕБГ,ЬРЁКШБЯЗБЛ	исключение
3.7	Максимальный сдвиг	Я	РЩДЧЫДШЕЬСЗЛЮВЙЗФУПЯМХТЖРЙЗФАТКНЙГЯБЪОДИЦЯЭЁЯЙ	СЪЕШЬЕЩЁЭТИМЯГКИХФРАНЦУЗСКИХБУЛОКДАВЫПЕЙЧАЮЖАК

Код программы для тестов:

```
    #include <unittest++/UnitTest++.h>
    #include <locale>
    #include
"/home/asic27/Timp/OOO_MOYA_OBORONA/modAlphaCipher.cpp"
    #include
"/home/asic27/Timp/OOO_MOYA_OBORONA/modAlphaCipher.h"
    struct KeyB_fixture {
        modAlphaCipher * p;
        KeyB_fixture() {
            std::wstring s;
            std::wstring_convert<std::codecvt_utf8<wchar_
t>, wchar_t> codec;
            s = codec.from_bytes("Б");
            p = new modAlphaCipher(s);
        }
        ~KeyB_fixture() {
            delete p;
        }
    };
    SUITE(KeyTest)
    {
         std::wstring_convert<std::codecvt_utf8<wchar_t>,
wchar_t> codec;
        TEST(ValidKey) {
            CHECK_EQUAL("ББГББ",modAlphaCipher(codec.from
_bytes("ББГ")).encrypt(codec.from_bytes("AAAAA")));
        }
        TEST(LongKey) {
```

```
            CHECK_EQUAL("БВГДЕ",modAlphaCipher(codec.from
_bytes("БВГДЕЖЗИК")).encrypt(codec.from_bytes("AAAAA")));
        }
        TEST(LowCaseKey) {
            CHECK_EQUAL("БВГБВ",modAlphaCipher(codec.from
_bytes("бвг")).encrypt(codec.from_bytes("AAAAA")));
        }
        TEST(DigitsInKey) {
                                CHECK_THROW(modAlphaCipher
cp(codec.from_bytes("B1")),cipher_error);
        }
        TEST(PunctuationInKey) {
                                CHECK_THROW(modAlphaCipher
cp(codec.from_bytes("B,C")),cipher_error);
        }
        TEST(WhitespaceInKey) {
                                CHECK_THROW(modAlphaCipher
cp(codec.from_bytes("B C")),cipher_error);
        }
        TEST(EmptyKey) {
                                CHECK_THROW(modAlphaCipher
cp(codec.from_bytes("")),cipher_error);
        }
        TEST(WeakKey) {
                                CHECK_THROW(modAlphaCipher
cp(codec.from_bytes("AAA")),cipher_error);
        }
    }
    SUITE(EncryptTest)
    {
         std::wstring_convert<std::codecvt_utf8<wchar_t>,
wchar_t> codec;
        TEST_FIXTURE(KeyB_fixture, UpCaseString) {
            CHECK_EQUAL("ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМП
ЛЕБГЬРЁКШБЯЗБЛ",p-
>encrypt(codec.from_bytes("СЪЕШЬЕЩЁЭТИМЯГКИХФРАНЦУЗСКИХБУ
ЛОКДАВЫПЕЙЧАЮЖАК")));
        }
        TEST_FIXTURE(KeyB_fixture, LowCaseString) {
            CHECK_EQUAL("ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМП
ЛЕБГЬРЁКШБЯЗБЛ",p-
>encrypt(codec.from_bytes("съешьещёэтимягкихфранцузскихбу
локдавыпейчаюжак")));
        }
```

```
                                    TEST_FIXTURE(KeyB_fixture,
StringWithWhitspaceAndPunct) {
            CHECK_EQUAL("ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМП
ЛЕБГЬРЁКШБЯЗБЛ",p->encrypt(codec.from_bytes("Съешь    ещё
эти мягких французских булок,да выпей чаю, жак")));
        }
        TEST_FIXTURE(KeyB_fixture, StringWithNumbers) {
            CHECK_EQUAL("ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФМП
ЛЕБГЬРЁКШБЯЗБЛ", p->encrypt(codec.from_bytes("1Съешь 2ещё
3эти 4мягких 5французских 6булок, 7да 8выпей 9чаю,
жак")));
        }
        TEST_FIXTURE(KeyB_fixture, EmptyString) {
                                            CHECK_THROW(p-
>encrypt(codec.from_bytes("")),cipher_error);
        }
        TEST_FIXTURE(KeyB_fixture, NoAlphaString) {
                                            CHECK_THROW(p-
>encrypt(codec.from_bytes("1234+8765=9999")),cipher_error
);
        }
        TEST(MaxShiftKey) {
            CHECK_EQUAL("РЩДЧЫДШЕЬСЗЛЮВЙЗФУПЯМХТЖРЙЗФАТКН
ЙГЯБЪОДИЦЯЭЁЯЙ",modAlphaCipher(codec.from_bytes("Я")).enc
rypt(codec.from_bytes("СЪЕШЬЕЩЁЭТИМЯГКИХФРАНЦУЗСКИХБУЛОКД
АВЫПЕЙЧАЮЖАК")));
        }
    }
    SUITE(DecryptText)
    {
         std::wstring_convert<std::codecvt_utf8<wchar_t>,
wchar_t> codec;
        TEST_FIXTURE(KeyB_fixture, UpCaseString) {
            CHECK_EQUAL("СЪЕШЬЕЩЁЭТИМЯГКИХФРАНЦУЗСКИХБУЛО
КДАВЫПЕЙЧАЮЖАК",p-
>decrypt(codec.from_bytes("ТЫЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФ
МПЛЕБГЬРЁКШБЯЗБЛ")));
        }
        TEST_FIXTURE(KeyB_fixture, LowCaseString) {
                                            CHECK_THROW(p-
>decrypt(codec.from_bytes("тыЁЩЭЁЪЖЮУЙНАДЛЙЦХСБОЧФИТЛЙЦВФ
МПЛЕБГЬРЁКШБЯЗБЛ")),cipher_error);
        }
        TEST_FIXTURE(KeyB_fixture, WhitespaceString) {
```

```
                CHECK_THROW(p->decrypt(codec.from_bytes("ТЫ
ЁЩЭЁЪЖЮУЙН                                АДЛЙЦХСБОЧФИТЛЙЦВФМ
ПЛЕБГЬРЁКШБЯЗБЛ")),cipher_error);
        }
        TEST_FIXTURE(KeyB_fixture, DigitsString) {
                                                CHECK_THROW(p-
>decrypt(codec.from_bytes("ТЫЁЩЭ2ЁЪЖЮУЙНАДЛЙЦ1ХСБОЧФИТЛЙЦ
ВФМПЛЕБГЬ4РЁКШБЯЗБ5Л")),cipher_error);
        }
        TEST_FIXTURE(KeyB_fixture, PunctString) {
                                                CHECK_THROW(p-
>decrypt(codec.from_bytes("ТЫЁЩ,ЭЁЪЖЮУЙНАДЛЙЦ,ХСБОЧФИТЛЙЦ
ВФМПЛЕБГ,ЬРЁКШБЯЗБЛ")),cipher_error);
        }
        TEST_FIXTURE(KeyB_fixture, EmptyString) {
                                                CHECK_THROW(p-
>decrypt(codec.from_bytes("")),cipher_error);
        }
        TEST(MaxShiftKey) {
            CHECK_EQUAL("СЪЕШЬЕЩЁЭТИМЯГКИХФРАНЦУЗСКИХБУЛО
КДАВЫПЕЙЧАЮЖАК",
                        modAlphaCipher(codec.from_bytes("
Я")).decrypt(codec.from_bytes("РЩДЧЫДШЕЬСЗЛЮВЙЗФУПЯМХТЖРЙ
ЗФАТКНЙГЯБЪОДИЦЯЭЁЯЙ")));
        }
    }
    int main(int argc, char **argv)
    {
        return UnitTest::RunAllTests();
    }
```

На рисунке 1 представлены результаты тестирования.

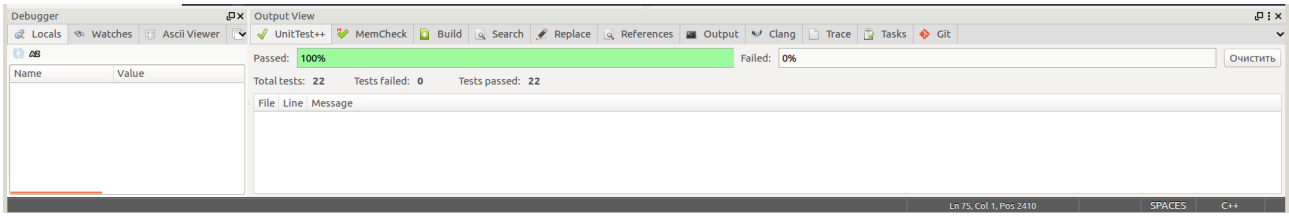

Рисунок 1 — Результаты тестов Гронсвельда на русском языке.

3.2 Были разработаны тестовые сценарии для программы, реализующей шифр маршрутной перестановки.

Таблица 4. - Тестовый сценарий для конструктора в программе, реализующей шифр маршрутной перестановки.

№	Тест	Параметры конструктора	Параметры метода shifr	Ожидаемый результат(конструктор)	Ожидаемый результат(shifr)
1.1	Верный ключ	3	BRAVENEW WORLD	-	ANWLREWR BVEOD
1.2	Плохой ключ	1	BRAVENEW WORLD	исключение	-

Таблица 5. - Тестовый сценарий для метода shifr.

№	Тест	Ключ	Параметры метода shifr	Ожидаемый результат shifr
2.1	Строка из прописных букв	3	BRAVENEWWORLD	ANWLREWRBVEOD
2.2	Строка из строчных букв	3	bravenewworld	ANWLREWRBVEOD
2.3	Строк с пробелами	3	BRAVE NEW WORLD	ANWLREWRBVEOD
2.4	Строка с цифра	3	2BRAVENEWWORLD1	ANWLREWRBVEOD
2.5	Пустая строка	3		исключение
2.6	Короткая строка	3	BRA	исключение
2.7	Строка со знаками препинания	3	BRAVENEWWORLD!!!	ANWLREWRBVEOD

Таблица 6. - Тестовый сценарий метода rashifr.

№	Тест	Ключ	Параметры метода rashifr	Ожидаемый результат rashifr
2.1	Строка из прописных букв	3	ANWLREWRBVEOD	BRAVENEWWORLD
2.2	Строка со строчными буквами	3	anwLREWRBVEOD	исключение
2.3	Строк с пробелами	3	ANWLR EWRBV EOD	исключение
2.4	Строка с цифра	3	1ANWLREWRBVEOD2	исключение
2.5	Пустая строка	3		исключение
2.6	Строка со знаками препинания	3	ANWLREWRBVEOD!!	исключение

2.7	Только цифры	3	1234	
2.8	Короткая строка	3	ANW	исключение

3.3 Были разработаны модульные тесты для тестирования программы, реализующей метод табличной маршрутной перестановки и проведено тестирование. Код тестов:

```
    #include <unittest++/UnitTest++.h>
    #include
"/home/asic27/Timp/Perestanovka/Perestanovka.cpp"
    #include
"/home/asic27/Timp/Perestanovka/Perestanovka.h"
    struct KeyThree_fixture {
        Perestanovka * p;
        KeyThree_fixture() {
            p = new Perestanovka(3);
        }
        ~KeyThree_fixture() {
            delete p;
        }
    };
    SUITE(KeyTest){
        TEST(ValidKey){
            CHECK_EQUAL("ANWLREWRBVEOD",Perestanovka(3).s
hifr("BRAVENEWWORLD"));
        }
        TEST(BadKey){
            CHECK_THROW(Perestanovka(1).shifr("BRAVENEWWO
RLD"),cipher_error);
        }
    }
    SUITE(ShifrTest){
        TEST_FIXTURE(KeyThree_fixture,UpCaseOT){
                            CHECK_EQUAL("ANWLREWRBVEOD",p-
>shifr("BRAVENEWWORLD"));
        }
        TEST_FIXTURE(KeyThree_fixture,LowCaseOT){
                            CHECK_EQUAL("ANWLREWRBVEOD",p-
>shifr("bravenewworld"));
        }
        TEST_FIXTURE(KeyThree_fixture,OTWithSpace){
              CHECK_EQUAL("ANWLREWRBVEOD",p->shifr("BRAVE
NEW WORLD"));
        }
```

```
        TEST_FIXTURE(KeyThree_fixture,OTWhithDigit){
                                CHECK_EQUAL("ANWLREWRBVEOD",p-
>shifr("2BRAVENEWWORLD1"));
        }
        TEST_FIXTURE(KeyThree_fixture,OTOnlyDigit){
            CHECK_THROW(p->shifr("1234"),cipher_error);
        }
        TEST_FIXTURE(KeyThree_fixture,OTPunct){
                                CHECK_EQUAL("ANWLREWRBVEOD",p-
>shifr("BRAVENEWWORLD!!!"));
        }
        TEST_FIXTURE(KeyThree_fixture,EmptyCT){
            CHECK_THROW(p->shifr(""),cipher_error);
        }
        TEST_FIXTURE(KeyThree_fixture,ShortOT){
            CHECK_THROW(p->shifr("BRA"),cipher_error);
        }
    }
    SUITE(RashifrTest){
        TEST_FIXTURE(KeyThree_fixture,UpCaseCT){
                                CHECK_EQUAL("BRAVENEWWORLD",p-
>rashifr("ANWLREWRBVEOD"));
        }
        TEST_FIXTURE(KeyThree_fixture,LowCaseCT){
                                                CHECK_THROW(p-
>rashifr("anwLREWRBVEOD"),cipher_error);
        }
        TEST_FIXTURE(KeyThree_fixture,CTwithSpace){
                        CHECK_THROW(p->rashifr("ANWLR  EWRBV
EOD"),cipher_error);
        }
        TEST_FIXTURE(KeyThree_fixture,CTwithdigits){
                                                CHECK_THROW(p-
>rashifr("1ANWLREWRBVEOD2"),cipher_error);
        }
        TEST_FIXTURE(KeyThree_fixture,EmptyCT){
            CHECK_THROW(p->rashifr(""),cipher_error);
        }
        TEST_FIXTURE(KeyThree_fixture,CTonlydigits){
            CHECK_THROW(p->rashifr("1234"),cipher_error);
        }
        TEST_FIXTURE(KeyThree_fixture,CTwithPunct){
                                                CHECK_THROW(p-
>rashifr("ANWLREWRBVEOD!"),cipher_error);
```

```
    }
    TEST_FIXTURE(KeyThree_fixture,ShortCT){
        CHECK_THROW(p->rashifr("ANW"),cipher_error);
    }

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

Результаты тестирования изображены на рисунке 2.

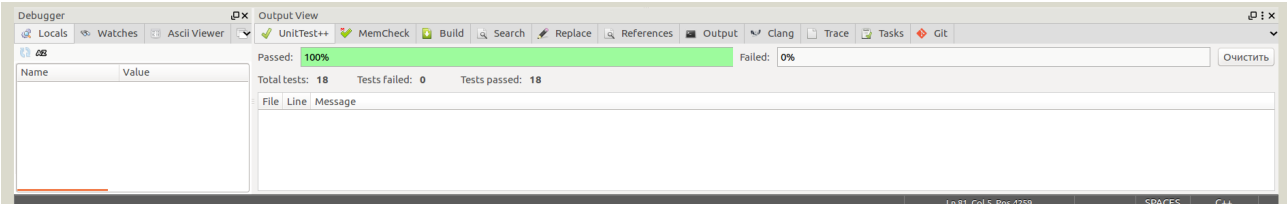

Рисунок 2 — Результаты тестирования программы, реализующей шифр маршрутной перестановки.

4. Вывод

В результате выполнения лабораторной работы было освоен процесс модульного тестирования разрабатываемых программ, были изучены возможности фреймворка UnitTest++ для проведения модульного тестирования. Были получены практические навыки по использованию макросов TEST, TEST_FIXTURE, CHECK_EQUALE, CHECK_THROW.