

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

ОТЧЁТ  
о лабораторной работе №2  
Обработка ошибок

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 обработки ошибок в программах на основе механизма исключений.

## 2 Задания к практической работе

2.1 Добавить к модулю шифрования русскоязычных сообщений методом Гронсвельда, разработанному при выполнении предыдущей работы, обработку исключений.

2.2 Добавить к модулю шифрования методом маршрутной перестановки, разработанной при выполнении предыдущей работы, обработку исключений.

## 3 Результат выполнения работы

3.1 К модулю шифрования шифром Гронсфельда на русском языке был добавлен обработчик ошибок, для этого в заголовочный файл `modAlphaCipher.h` был добавлен класс `cipher_error` как производный класс от стандартного `invalid_argument`. Добавленный класс не отличается от класса, данного в методических указаниях в качестве примера. В закрытой секции класса `modAlphaCipher` были объявлены три новых метода: `wstring getValidKey(const wstring & s)`-для проверки ключа, `wstring getValidOpenText(const wstring & s)`-для проверки открытого текста, который необходимо зашифровать и `wstring getValidCipherText(const wstring & s)`-для проверки текста, требующего расшифровки. Данные методы получают в качестве параметра и возвращают широкие строки.

Метод `getValidKey`:

```
inline std::wstring modAlphaCipher::getValidKey(const
std::wstring & s)
{   std::locale loc("ru_RU.UTF-8");
    std::wstring_convert<std::codecvt_utf8<wchar_t>,
wchar_t> codec;
    if (s.empty())
        throw cipher_error("Empty key");
    std::wstring tmp(s);
    for (auto & c:tmp) {
```

```

        if ((c<L'A' || c>L'я') and c!=L'Ё' and c!
='ё') {
            std::string t;
            t = codec.to_bytes(s);
            throw cipher_error(std::string("Invalid
key ") + t);
        }
        if (iswlower(c))
            c = towupper(c);
    }
    return tmp;
}

```

Данный метод отличается от метода приведённого в качестве примера в методических указаниях использованием широких строк и проверкой на букву. Если проверка на принадлежность к русскому алфавиту не пройдена, тогда полученная на вход строка `s` типа `wstring`, переводится в `string`, записывается в новую другую строку, которая после передаётся в качестве параметра `cipher_error`. Перевод в `string` необходим, так как `cipher_error` не работает с `wstring`.

Метод `getValidOpenText`:

```

inline std::wstring
modAlphaCipher::getValidOpenText(const std::wstring & s)
{
    std::wstring tmp;
    for (auto c:s) {
        if ((c>=L'A' && c<=L'я') || c==L'Ё' ||
c=='ё') {
            if (iswlower(c))
                tmp.push_back(towupper(c));
            else
                tmp.push_back(c);
        }
    }
    if (tmp.empty())
        throw cipher_error("Empty open text");
    return tmp;
}

```

В данном методе была изменена лишь проверка на русскую букву, а также функции для проверки регистра и перевода регистра.

Метод `getValidCipherText`:

```
inline                                     std::wstring
modAlphaCipher::getValidCipherText(const std::wstring &
s)
{
    std::locale loc("ru_RU.UTF-8");
    std::wstring_convert<std::codecvt_utf8<wchar_t>,
wchar_t> codec;
    if (s.empty())
        throw cipher_error("Empty cipher text");
    for (auto c:s) {
        if ((c<L'A' || c>L'Я') and c!=L'Ё') {
            std::string t;
            t = codec.to_bytes(s);
            std::cout<<t;
            throw cipher_error(std::string("Invalid
cipher text ") + t);
        }
    }
    return s;
}
```

В методе была изменена проверка на русский символ, а также перевод строки `wstring` в `string`, так как в случае обнаружения не буквы, строку нужно передавать в `cipher_error`. На рисунках 1, 2, 3, 4, 5, 6 изображены результаты работы программы с различными входными данными.



```
Терминал
Cipher ready. Input key: Эхо
Key loaded
Cipher ready. Input operation (0-exit, 1-encrypt, 2-decrypt): 1
Cipher ready. Input text: ПРИВЕТ
Encrypted text: МЕЧЯЬБ
Cipher ready. Input operation (0-exit, 1-encrypt, 2-decrypt): █
```

Рисунок 1 — Нижний регистр в ключе.

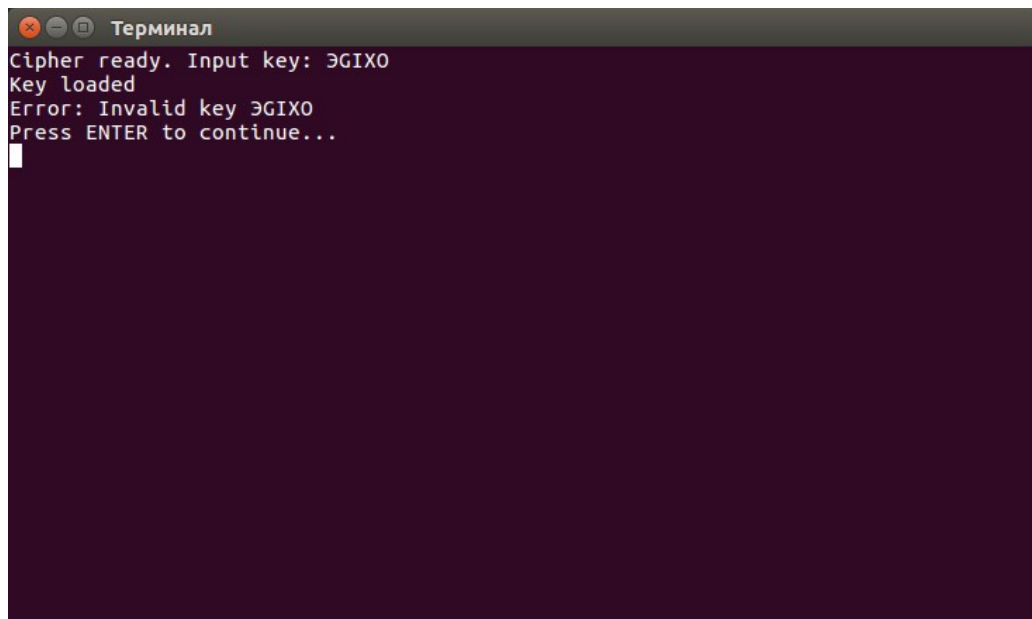


Рисунок 2 — Посторонние символы в ключе.

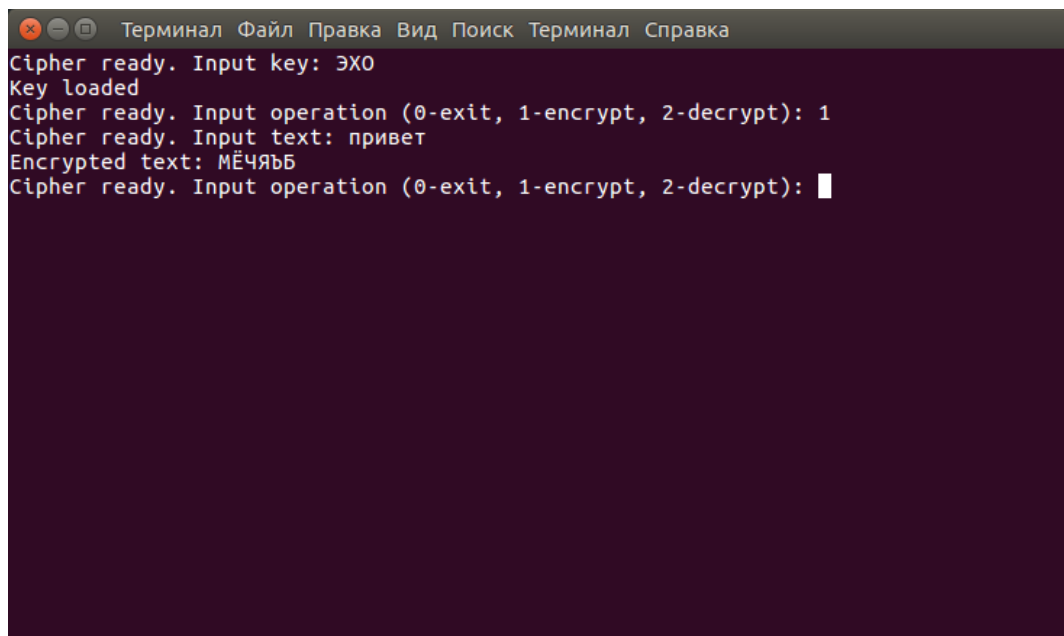


Рисунок 3 — Буквы нижнего регистра в открытом тексте.

```
Терминал
Cipher ready. Input key: ЭХО
Key loaded
Cipher ready. Input operation (0-exit, 1-encrypt, 2-decrypt): 1
Cipher ready. Input text: ABCDIFGHIJKLMNOPQRSTUVWXYZsasa
Encrypted text: МЁЧЯЬБ
Cipher ready. Input operation (0-exit, 1-encrypt, 2-decrypt):
```

Рисунок 4 — Не русские буквы в открытом тексте.

```
Терминал
Cipher ready. Input key: ЭХО
Key loaded
Cipher ready. Input operation (0-exit, 1-encrypt, 2-decrypt): 2
Cipher ready. Input text: МЁЧЯЬБ
Decrypted text: ПРИВЕТ
Cipher ready. Input operation (0-exit, 1-encrypt, 2-decrypt):
```

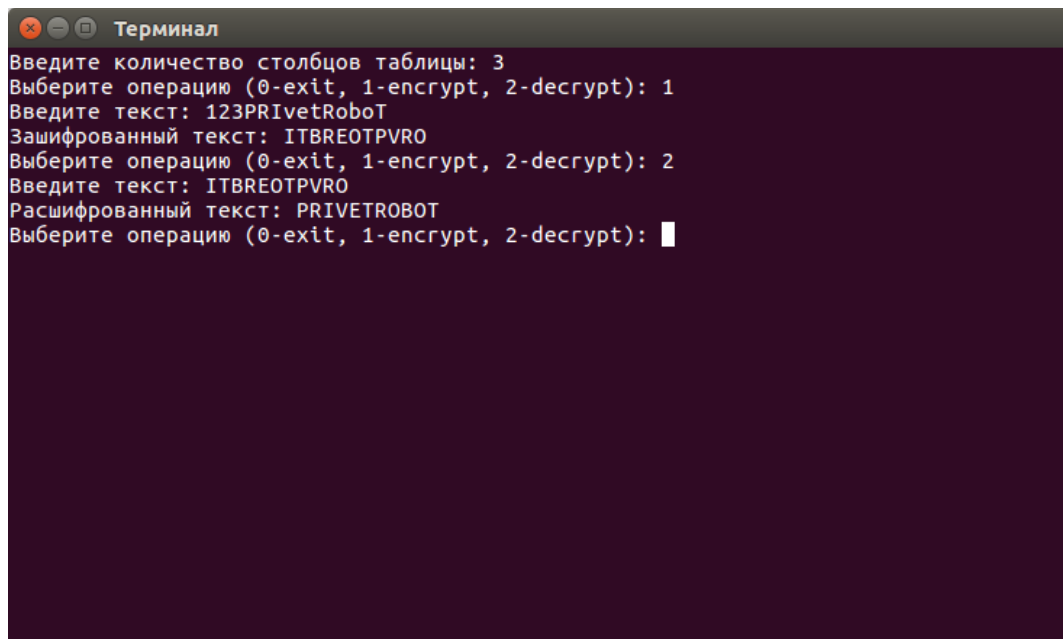
Рисунок 5 — Правильный текст для расшифровки.



```
Терминал
Cipher ready. Input key: ЭХ0
Key loaded
Cipher ready. Input operation (0-exit, 1-encrypt, 2-decrypt): 2
Cipher ready. Input text: мЁчяьб
Error: Invalid cipher text мЁчяьб
Press ENTER to continue...
```

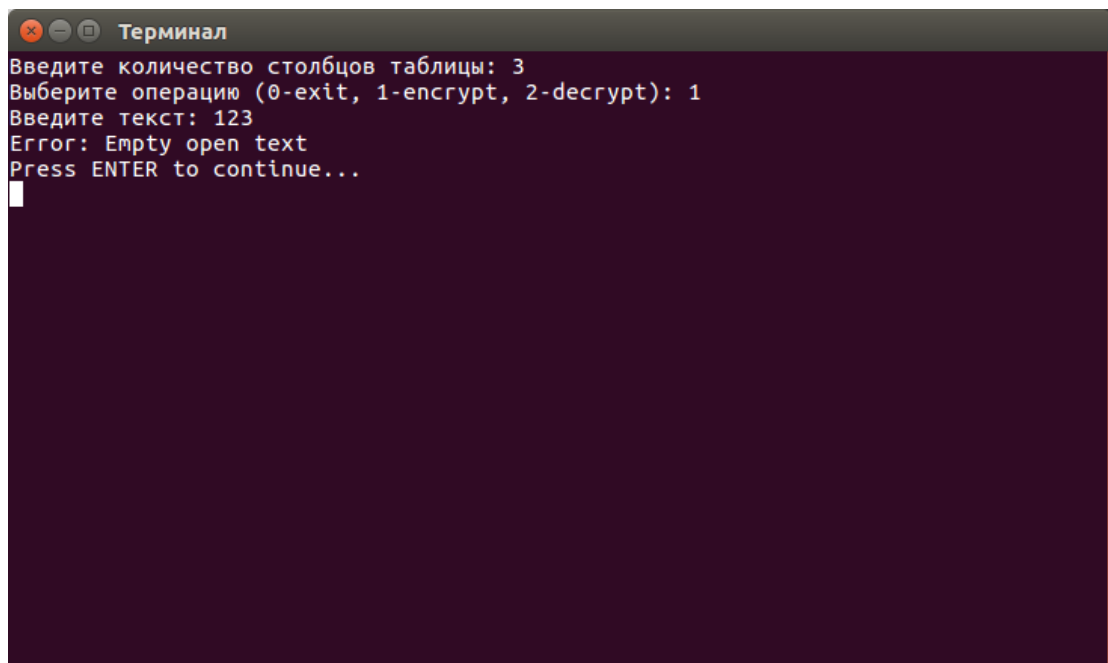
Рисунок 6 — Буквы в нижнем регистре.

3.2 В программу, реализующую шифр маршрутной табличной перестановки была добавлен обработка исключений. В заголовочный файл `Perestanovka.h` был добавлен класс `cipher_error` как производный класс от стандартного `invalid_argument`. Добавленный класс не отличается от класса, данного в методических указаниях в качестве примера. В класс `Perestanonvka` были добавлены методы для проверки открытого текста и расшифрованного текста. Методы для проверки не отличаются от тех, что приведены в методических указаниях к лабораторной работе. В методах `rashifr` и `shifr` добавлены строки: `std::string m=getValidCipherText(z), std::string m=getValidOpenText(t)` для того, чтобы перед шифрованием или расшифровкой текст проверялся. На рисунках 7, 8, 9 изображены результаты работы программы в различных случаях.



```
Терминал
Введите количество столбцов таблицы: 3
Выберите операцию (0-exit, 1-encrypt, 2-decrypt): 1
Введите текст: 123PRIVetRoboT
Зашифрованный текст: ITBRE0TPVRO
Выберите операцию (0-exit, 1-encrypt, 2-decrypt): 2
Введите текст: ITBRE0TPVRO
Расшифрованный текст: PRIVETROBOT
Выберите операцию (0-exit, 1-encrypt, 2-decrypt):
```

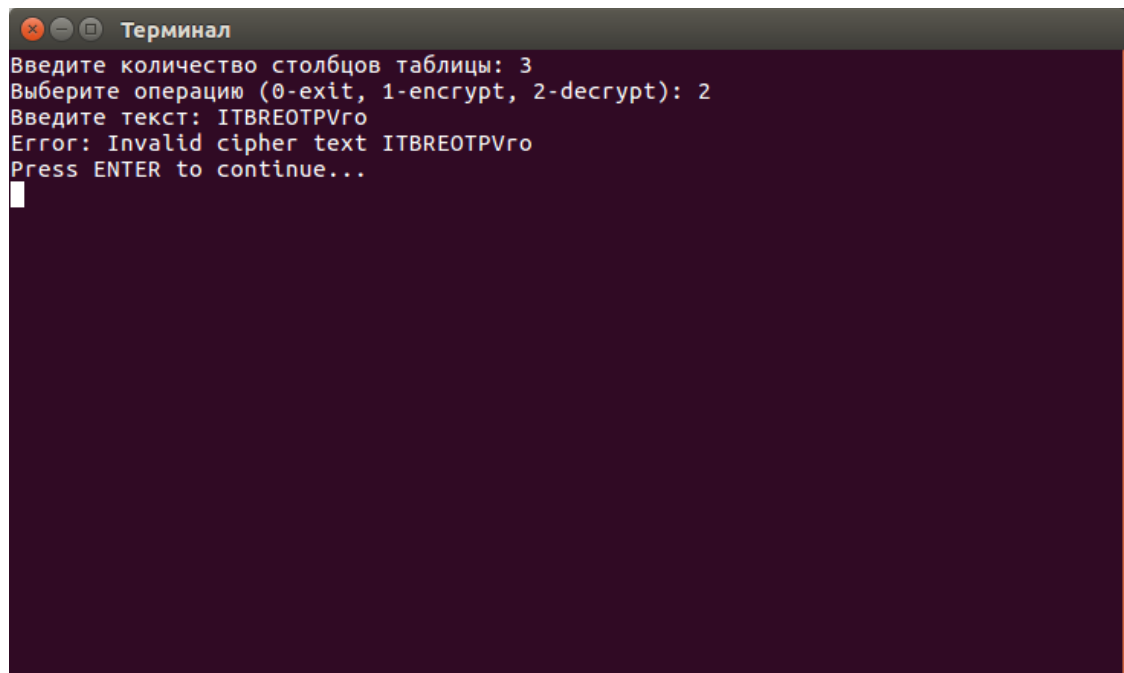
Рисунок 7 — Открытый текст с посторонними символами.



```
Терминал
Введите количество столбцов таблицы: 3
Выберите операцию (0-exit, 1-encrypt, 2-decrypt): 1
Введите текст: 123
Error: Empty open text
Press ENTER to continue...
```

Рисунок 8 — Открытый текст из посторонних символов.





```
Терминал
Введите количество столбцов таблицы: 3
Выберите операцию (0-exit, 1-encrypt, 2-decrypt): 2
Введите текст: ITBRE0TPVro
Error: Invalid cipher text ITBRE0TPVro
Press ENTER to continue...
```

Рисунок 9 — Расшифровываемый текст с посторонними символами.

#### 4. Вывод

В процессе выполнения лабораторной работы были изучены возможности языка C++ по обработке ошибок в программе на основе механизма исключений, была освоена обработка исключений, были получены практические навыки по применению try-catch.