МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационная безопасность систем и технологий»

Отчет
о лабораторной работе №2
на тему «Обработка ошибок»

Дисциплина: Методы и средства программирования
Группа: 22ПИ2
Выполнил: Никитина М. А.
Количество баллов:
Дата сдачи:
Принял: Н. А. Сидоров

2023

1 Цель работы: освоить процесс обработки ошибок в программах на основе механизма исключений.

2 Задание на лабораторную работу

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 Порядок выполнения работы

3.1 Была добавлена к модулю шифрования русскоязычных сообщений методом Гронсвельда, разработанному при выполнении предыдущей работы, обработка исключений. Результат представлен на рисунке 1. Код программы представлен в приложении А.

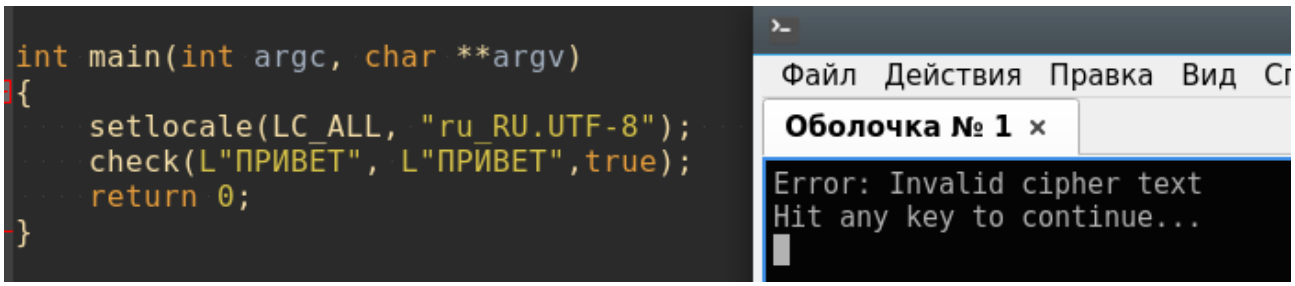

Рисунок 1 - Вывод ошибки

3.2 Была добавлена к модулю шифрования методом маршрутной перестановки, разработанной при выполнении предыдущей работы, обработка исключений. Результат работы представлен на рисунке 2. Код программы представлен в приложении Б.

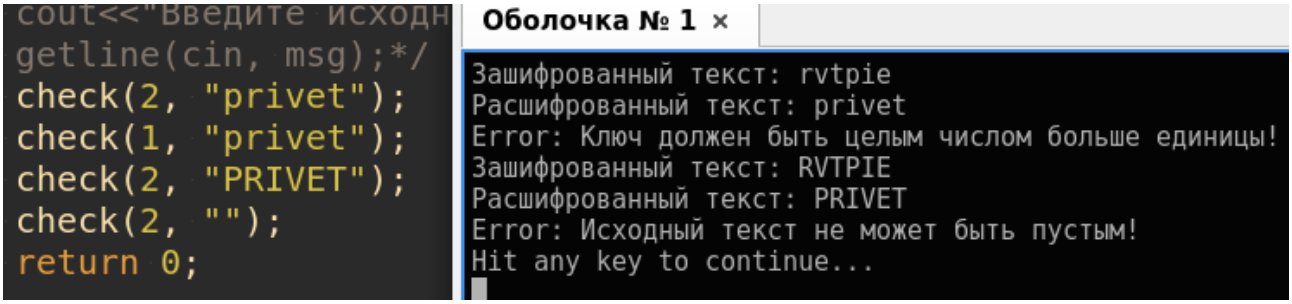

Рисунок 2 - Проверка вариантов

4 Выводы

С помощью данной лабораторной работы был освоен процесс создания многомодульных проектов. Полный результат работы можно посмотреть по ссылке: https://github.com/Popitka994/MiSP/Lb2.

Приложение А

Код модуля main.cpp:

```
#include <iostream>
#include "modAlphaCipher.h"
#include <locale>
#include <clocale>
#include <cwctype>
#include <cctype>
using namespace std;
void check(const wstring& Text, const wstring& key, const
bool destructCipherText=false)
{
    try {
        setlocale(LC_ALL, "ru_RU.UTF-8");
        wstring cipherText;
        wstring decryptedText;
        modAlphaCipher cipher(key);
        cipherText = cipher.encrypt(Text);
        if(destructCipherText) cipherText+=L'и';
        decryptedText = cipher.decrypt(cipherText);
   wcout<<"key="<<key<<endl;
        wcout<<Text<<endl;
        wcout<<cipherText<<endl;
        wcout<<decryptedText<<endl;
    } catch (const cipher_error & e) {
        cerr<<"Error: "<<e.what()<<endl;
    }
}
```

```
int main(int argc, char **argv)
{
    setlocale(LC_ALL, "ru_RU.UTF-8");
    check(L"ПРИВЕТ", L"ПРИВЕТ",true);
    return 0;
}
```

Код модуля modAlphaCipher.cpp:

```
#include "modAlphaCipher.h"
#include <locale>
using namespace std;
modAlphaCipher::modAlphaCipher(const std::wstring& skey)
{
        for (unsigned i=0; i<numAlpha.size(); i++)
alphaNum[numAlpha[i]]=i;
    key = convert(getValidKey(skey));
}
wstring modAlphaCipher::encrypt(const std::wstring&
open_text) {
                        std::vector<int> work =
convert(getValidOpenText(open_text));
     for(unsigned i=0; i < work.size(); i++) work[i] =
(work[i] + key[i % key.size()]) % alphaNum.size();
    return convert(work);
}
wstring modAlphaCipher::decrypt(const std::wstring&
cipher_text) {
                        std::vector<int> work =
convert(getValidCipherText(cipher_text));
```

```
     for(unsigned i=0; i < work.size(); i++) work[i] = (work[i] + alphaNum.size() - key[i % key.size()]) % alphaNum.size();
    return convert(work);
}
inline std::vector<int> modAlphaCipher::convert(const std::wstring& s) {
    std::vector<int> result;
    for(auto c:s) result.push_back(alphaNum[c]);
    return result;
}
inline std::wstring modAlphaCipher::convert(const std::vector<int>& v) {
    std::wstring result;
    for(auto i:v) result.push_back(numAlpha[i]);
    return result;
}
inline std::wstring modAlphaCipher::getValidKey(const std::wstring & s) {
    if (s.empty()) throw cipher_error("Empty key");
    std::wstring tmp(s);
    for (auto c:tmp) {
                              if (!iswalpha(c)) throw cipher_error(std::string("Invalid key"));
        if (iswlower(c)) c = towupper(c);
    }
 return tmp;
}
```

```
inline                                          std::wstring
modAlphaCipher::getValidOpenText(const std::wstring & s)
{
    std::wstring tmp;
    for (auto c:s) {
        if (iswalpha(c)) {
            if (iswlower(c)) tmp.push_back(towupper(c));
            else tmp.push_back(c);
        }
    }
       if (tmp.empty()) throw cipher_error("Empty open
text");
    return tmp;
}
inline  std::wstring  modAlphaCipher::getValidCipherText
(const std::wstring & s) {
       if (s.empty()) throw cipher_error("Empty cipher
text");
          for  (auto  c:s)  if  (!iswupper(c))  throw
cipher_error(std::string("Invalid cipher text "));
    return s;
}
```

Код модуля modAlphaCipher.h:

```
#pragma once
#include <vector>
#include <string>
#include <map>
#include <locale>
class modAlphaCipher
```

```
{
private:
                        std::wstring      numAlpha      =
L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФКЦЧШЩЪЫЬЭЮЯ";     //алфавит     по
порядку
       std::map  <wchar_t,int>  alphaNum;  //ассоциативный
массив "номер по символу"
    std::vector <int> key; //ключ
    std::vector<int> convert(const std::wstring& s);
    std::wstring convert(const std::vector<int>& v);
    std::wstring getValidKey(const std::wstring & s);
      std::wstring  getValidOpenText(const  std::wstring  &
s);
     std::wstring getValidCipherText(const std::wstring &
s);
public:
    modAlphaCipher()=delete;
    modAlphaCipher(const std::wstring& skey);
    std::wstring encrypt(const std::wstring& open_text);
             std::wstring   decrypt(const   std::wstring&
cipher_text);
};
class cipher_error: public std::invalid_argument {
public:
    explicit cipher_error (const std::string& what_arg):
    std::invalid_argument(what_arg) {}
    explicit cipher_error (const char* what_arg):
    std::invalid_argument(what_arg) {}
};
```

Приложение Б

Код программы для модуля main.cpp:

```
#include <iostream>
#include "cipher.h"
using namespace std;

void check (int key, const string& msg)
{
    try {
          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);
        string encrypted=cipher.encrypt(msg);
        string decrypted=cipher.decrypt(encrypted);
        cout<<"Зашифрованный текст: "<<encrypted<<endl;
        cout<<"Расшифрованный текст: "<<decrypted<<endl;
    } catch (const exception& e) {
        cerr<<"Error: "<<e.what()<<endl;
    }
}

int main ()
{
    string msg;
    int key;
    check(2, "privet");
    check(1, "privet");
```

```
    check(2, "PRIVET");
    check(2, "");
    return 0;
}
```

Код программы для модуля cipher.cpp:

```
#include "cipher.h"
#include <string>
using namespace std;

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

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;
}

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 программы для модуля cipher.h:

```
#ifndef CIPHER_H
```

```
#define CIPHER_H
#include <string>
using namespace std;
class Cipher {
private:
    int key;
public:
    Cipher(int k);
    string encrypt(const string& msg);
string decrypt(const string& zashifrovan);
};
#endif
```