Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Дисциплина: «Защита информации»

Профиль: «Программная инженерия»

Семестр 5

ОТЧЕТ

по лабораторной работе №1

Тема: «Шифры перестановки и замены»

Выполнил: студент группы РИС-19-1б

Миннахметов Э.Ю. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Шереметьев В. Г. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

Пермь, 2021

**ЦЕЛЬ РАБОТЫ**

Получить практические навыки по применению шифров перестановки и шифров простой замены.

**ЗАДАНИЕ**

Реализовать шифрование текстового сообщения, используя шифр Гронсфельда.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Шифр Гронсфельда** — полиалфавитный подстановочный шифр создан графом Гронсвельдом (руководителем первой дешифровальной службы Германии) в XVII веке. Шифр можно считать усовершенствованием шифра Цезаря (надежность) и Виженера / Бофора (скорость).



Рисунок 1 – Таблица Гронсфельда

Длина ключа (*K*) должна быть равной длине исходного текста. Для этого циклически записывают ключ до тех пор, пока его длина не будет соответствовать длине исходного текста.

**Шифрование**. Каждый символ *Mi* открытого текста *M* нужно на *Ki* (соответствующий символ ключа *K*) шагов сдвинуть вправо. Или пользуясь таблицей Гронсфельда (*Tx y*, где *x* — номер строки, а *y* — номер столбца и отсчет ведется с нуля): каждый символ *Ci* шифротекста *C* находится на пересечении столбца *y*, первый (заголовочный) символ которого равен соответствующему символу открытого текста *Mi*, и *Ki*-й (соответствующей цифры ключа) строки — (*TKi y*).

**Дешифрование**. Каждый символ (*Ci*) зашифрованного текста *C* нужно на *Ki* (соответствующий символ ключа *K*) шагов сдвинуть влево. Или пользуясь таблицей Гронсфельда (*Tx y*, где *x* — номер строки, а *y* — номер столбца и отсчет ведется с нуля): нужно в *Ki* (*i*-ая цифра ключа *K*) строке найти символ, который равен соответствующему символу шифротекста (*TKi y = Ci*), и первый (заголовочный) элемент столбца будет *i*-ый символ открытого текста.

**ХОД РАБОТЫ**

На рисунке 2 представлена форма, в первое поле которой вводится ключ шифрования, а во второе шифруемый текст, а в третьем выводится результат шифрования после нажатия на кнопку «Зашифровать». Если же требуется расшифровать, то шифротекст помещается в третье поле и уже требуется нажатие на кнопку «Расшифровать», а результат помещается во второе поле. Числа должны быть отделены между собой пробелом.

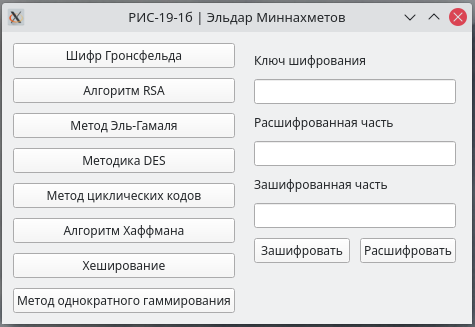


Рисунок 2 – Форма для шифрования.

Пример работы программы представлен на рисунке 3.

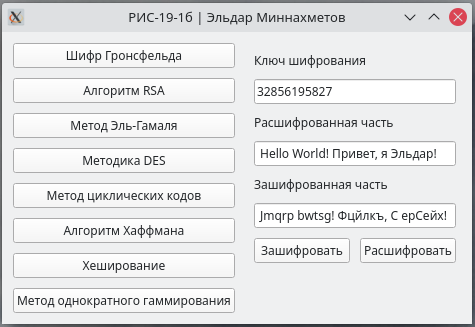


Рисунок 3 – Пример работы программы.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

class Code {  
private:  
 static QString alphabet;  
  
 static ushort get(int i) {  
 return alphabet[i].unicode();  
 }  
  
 static bool check(const QChar &c, const QCharRef &a, const QCharRef &z) {  
 ushort uc = c.unicode();  
 return a.unicode() <= uc && uc <= z.unicode();  
 }  
  
 static bool check(const QChar &c) {  
 return check(c, alphabet[0], alphabet[25]) ||  
 check(c, alphabet[26], alphabet[51]) ||  
 check(c, alphabet[52], alphabet[83]) ||  
 check(c, alphabet[84], alphabet[115]);  
 }  
  
 static void index(int &i, const QChar &c, int begin, int end) {  
 if(c >= get(begin) && c <= get(end)) {  
 i = c.unicode() - get(begin) + begin;  
 }  
 }  
  
 static QChar cry(const QChar &c, const int &term, bool decrypt = false) {  
 int i;  
 index(i, c, 0, 25);  
 index(i, c, 26, 51);  
 index(i, c, 52, 83);  
 index(i, c, 84, 115);  
 int temp = i + (decrypt ? -1 : 1) \* term;  
 int size = alphabet.size();  
 temp = temp < 0 ? temp + size : (temp >= size ? temp - size : temp);  
 return alphabet.constData()[temp];  
 }  
  
 std::vector<int> \_code;  
  
public:  
 static QString alph() { return alphabet; }  
  
 explicit Code(const QString &code) {  
 for(auto i : code) {  
 if(i >= QChar('0') && i <= QChar('9')) {  
 \_code.push\_back(i.unicode() - QChar('0').unicode());  
 }  
 }  
 }  
  
 QString crypt(const QString &str, bool decrypt = false) {  
 QString result;  
 for(int i = 0, j = 0; i < str.size(); ++i) {  
 result.append(check(str[i]) ? cry(str[i], \_code[j], decrypt) : str[i]);  
 ++j == \_code.size() ? j = 0 : j;  
 }  
 return result;  
 }  
  
 QString decrypt(const QString &str) {  
 return crypt(str, true);  
 }  
};

QString Code::alphabet = QString("ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
 "abcdefghijklmnopqrstuvwxyz"  
 "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"  
 "абвгдежзийклмнопрстуфхцчшщъыьэюя");