**Львівський національний університет імені Івана Франка**

**Факультет електроніки та комп’ютерних технологій**

*Кафедра радіоелектронних і комп’ютерних систем*

**Звіт**

про виконання лабораторної роботи №2

*«Програмна реалізація алгоритмів шифрування класичними криптосистемами»*

**Виконав:**

студентки групи ФеІ-31

Зьола О.П.

Викладач:  
Сінькевич О.О.

Львів

2020

**Мета роботи:** навчитися застосовувати на практиці класичні методи шифрування текстової інформації, використовуючи засоби об’єктно – орієнтованої мови програмування.

**Завдання:** Реалізувати на практиці алгоритм шифрування текстової інформації за вибраними методами з використанням об’єктно – орієнтованої мови програмування.

**Порядок виконання роботи:**

1. Для програмної реалізації шифрування текстової інформації я обрала 4 наступні методи: *шифр простої підстановки, транспозиція з фіксованим періодом d, шифр Віженера* та *шифр* *Цезаря.* Як інструмент обрала об’єктно – орієнтовану мову програмування С++ та QT Framework.
2. Розробила алгоритми та написала відповідні методи шифрування текстової інформації. Протестувала реалізовані методи на наведених в методичних матеріалах прикладах.

**Текст програми:**

#include <QCoreApplication>

#include <iostream>

#include <QMap>

#include <QString>

#include <QDebug>

QString **Simple\_substitution\_cipher**(QString s, QString key = "XGUACDTBFHRSLMQVYZWIEJOKNP")

{

int len = s.size();

QString encrypted(len);

*static* QMap <char, QChar> map;

*for* (int i = 0; i < 26; i++)

{

map.insert('A'+i,key.at(i).toUpper());

map.insert('a'+i,key.at(i).toLower());

}

QList<QChar> values = map.values();

QList<char> keys = map.keys();

*for* (int i = 0; i < s.size(); i++)

{

*for* (int j = 0; j < values.size(); j++)

{

*if* (s[i] == keys[j])

encrypted[i] = values[j];

s.remove(i);

}

}

*return* encrypted;

}

QString **Simple\_substitution\_decryption** (QString s, QString key = "XGUACDTBFHRSLMQVYZWIEJOKNP")

{

int len = s.size();

QString decrypted(len);

*static* QMap <char, QChar> map ;

*for* (int i = 0; i < 26; i++)

{

map.insert('A'+i,key.at(i).toUpper());

map.insert('a'+i,key.at(i).toLower());

}

QList<QChar> values = map.values();

QList<char> keys = map.keys();

*for* (int i = 0; i < s.size(); i++)

{

*for* (int j = 0; j < values.size(); j++)

{

*if* (s[i] == values[j])

decrypted[i] = keys[j];

s.remove(i);

}

}

*return* decrypted;

}

QString **Caesar\_cipher**(QString s, int l)

{

int len = s.size();

QString encrypted(len);

int str\_ascii = 0;

*for* (int i = 0; i < s.size(); i++)

{

str\_ascii = s.at(i).toLatin1();

*if*(s.at(i).isUpper())

{

encrypted[i] = (str\_ascii + l - 'A') % 26 + 'A';

}

*if*(s.at(i).isLower())

{

encrypted[i] = (str\_ascii + l - 'a') % 26 + 'a';

}

}

*return* encrypted;

}

QString **Caesar\_decryption**(QString s, int l)

{

int len = s.size();

QString decrypted(len);

int str\_ascii = 0;

*for* (int i = 0; i < s.size(); i++)

{

str\_ascii = s.at(i).toLatin1();

*if*(s.at(i).isUpper())

{

decrypted[i] = (str\_ascii + 26 - l - 'A') % 26 + 'A' ;

}

*if*(s.at(i).isLower())

{

decrypted[i] = (str\_ascii + 26 - l - 'a') % 26 + 'a';

}

}

*return* decrypted;

}

QString **Transposition\_cipher**(QString s, int d = 23154)

{

QVector<int> key;

QString encrypted(s.size());

int number\_of\_digits = 0;

*while* (!(d/10 == 0))

{

number\_of\_digits++;

key.push\_front(d%10);

d/=10;

}

key.push\_front(d%10);

*for* (int i = 0; i < s.size(); i++)

{

encrypted[i] = s[key[i] - 1];

}

*return* encrypted;

}

QString **Transposition\_decryption**(QString s, int d = 23154)

{

QVector<int> key;

QString decrypted(s.size());

int number\_of\_digits = 0;

*while* (!(d/10 == 0))

{

number\_of\_digits++;

key.push\_front(d%10);

d/=10;

}

key.push\_front(d%10);

*for* (int i = 0; i < s.size(); i++)

{

decrypted[key[i] - 1] = s[i];

}

*return* decrypted;

}

QString **Vigenere\_cipher**(QString str, QString key)

{

int len = str.size();

QString encrypted(len);

int key\_ascii = 0, str\_ascii = 0;

int i = 0;

*while* (key.size() < str.size())

{

key.push\_back(key[i]);

i++;

}

*for* (int i = 0; i < str.size(); i++)

{

*if*(str.at(i).isUpper())

{

key\_ascii = key.at(i).toUpper().toLatin1();

str\_ascii = str.at(i).toUpper().toLatin1();

encrypted[i] = (key\_ascii + str\_ascii) % 26 + 65;

}

*if*(str.at(i).isLower())

{

key\_ascii = key.at(i).toLower().toLatin1();

str\_ascii = str.at(i).toLower().toLatin1();

encrypted[i] = (key\_ascii + str\_ascii) % 26 + 65;

}

}

*return* encrypted;

}

QString **Vigenere\_decryption**(QString str, QString key)

{

int len = str.size();

QString decrypted(len);

int key\_ascii = 0, str\_ascii = 0;

int i = 0;

*while* (key.size() < str.size())

{

key.push\_back(key[i]);

i++;

}

*for* (int i = 0; i < str.size(); i++)

{

*if*(str.at(i).isUpper())

{

key\_ascii = key.at(i).toUpper().toLatin1();

str\_ascii = str.at(i).toUpper().toLatin1();

decrypted[i] = (str\_ascii - key\_ascii + 26) % 26 + 65;

}

*if*(str.at(i).isLower())

{

key\_ascii = key.at(i).toLower().toLatin1();

str\_ascii = str.at(i).toLower().toLatin1();

decrypted[i] = (str\_ascii - key\_ascii + 26) % 26 + 65;

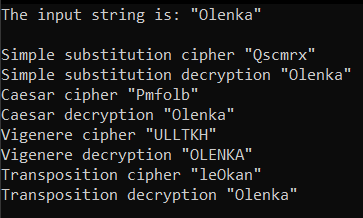
}

}

*return* decrypted;

}

**Результати роботи:**

****

***Скрін програми 1.*** *Результат роботи програми.*

**Висновок:** Виконуючи цю лабораторну роботу, я навчилася застосовувати на практиці класичні методи шифрування текстової інформації, використовуючи засоби об’єктно – орієнтованої мови програмування.