# **Лабораторная работа №4**

**Исследование криптографических шифров на основе подстановки(замены) символов**

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации подстановочных шифров.

**Задачи:**1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости подстановочных шифров.

2. Ознакомиться с особенностями реализации и свойствами  
различных подстановочных шифров на основе готового программного средства (L\_LUX).

3. Разработать приложение для реализации указанных преподавателем методов подстановочного зашифрования/расшифрования.

4. Выполнить исследование криптостойкости шифров на основе статистических данных о частотах появления символов в исходном и зашифрованном сообщениях.

5. Оценить скорость зашифрования/расшифрования реализованных способов шифров.

6. Результаты выполнения лабораторной работы оформить  
в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов  
эксперимента.

**Код программы**

import numpy as np  
import matplotlib.pyplot as plt  
  
def main():  
 keyCipher = "mazukevitch".upper()  
 message = "ribchinskiyantondmitrievicth"  
 print("----- Шифр Цезаря -----")  
 CaesarCipher(keyCipher, message.upper(), "latinica", 10)  
  
 print("\n----- Шифр Порты -----")  
 message = "АЛЕКСЕЙ "  
 PortCipher(message.upper(), "kirillica")  
  
  
def CaesarCipher(key, message, alphabet, shift):  
 if(alphabet == "latinica"):  
 start\_symbol = 65  
 end\_symbol = 91  
 power\_alphabet = 26  
 elif(alphabet == "kirillica"):  
 start\_symbol = 1040  
 end\_symbol = 1072  
 power\_alphabet = 32  
  
 print("Ключ:", key)  
 print("Сдвиг ключа:", shift)  
  
 original\_alphabet = np.zeros((power\_alphabet), dtype=str)  
 modified\_alphabet = np.zeros((power\_alphabet), dtype=str)  
  
 for letter in range(start\_symbol, end\_symbol, 1):  
 original\_alphabet[letter - start\_symbol] = chr(letter)  
  
 current\_index = 0  
 for letter in range(len(key)):  
 if(key[letter] not in modified\_alphabet):  
 if(current\_index + shift == power\_alphabet):  
 shift = 0  
 current\_index = 0  
 else:  
 modified\_alphabet[current\_index + shift] = key[letter]  
 current\_index += 1  
  
 current\_index += shift  
 for letter in range(start\_symbol, end\_symbol, 1):  
 if(chr(letter) not in modified\_alphabet):  
 if(current\_index == power\_alphabet):  
 shift = 0  
 current\_index = 0  
 if (modified\_alphabet[current\_index] == ""):  
 modified\_alphabet[current\_index] = chr(letter)  
 current\_index += 1  
 else:  
 current\_index += 1  
 else:  
 modified\_alphabet[current\_index] = chr(letter)  
 current\_index += 1  
  
  
 encrypted\_message = ""  
 for elem in range(len(message)):  
 encrypted\_message += modified\_alphabet[list(original\_alphabet).index(message[elem])]  
  
 np\_probability\_original\_alphabet = np.zeros((power\_alphabet), dtype=float)  
 for elem in range(len(original\_alphabet)):  
 for letter in message:  
 if(original\_alphabet[elem] == letter):  
 np\_probability\_original\_alphabet[elem] += 1  
 np\_probability\_original\_alphabet[elem] = round(np\_probability\_original\_alphabet[elem] / len(message), 3)  
  
 np\_probability\_modified\_alphabet = np.zeros((power\_alphabet), dtype=float)  
 for elem in range(len(original\_alphabet)):  
 for letter in message:  
 if (original\_alphabet[elem] == letter):  
 np\_probability\_modified\_alphabet[elem] += 1  
 np\_probability\_modified\_alphabet[elem] = round(np\_probability\_modified\_alphabet[elem] / len(message), 3)  
  
 crypted\_message = ""  
 for elem in range(len(encrypted\_message)):  
 crypted\_message += original\_alphabet[list(original\_alphabet).index(message[elem])]  
  
 print("Исходный алфавит:", original\_alphabet)  
 print("Модифицированный алфавит:", modified\_alphabet)  
 print("Входящее сообщение:", message)  
 print("Зашифрованное сообщение:", encrypted\_message)  
 print("Расшифрованное сообщение:", crypted\_message)  
  
 fig, ax = plt.subplots()  
 ax.bar(original\_alphabet, np\_probability\_original\_alphabet)  
 ax.set\_facecolor('seashell')  
 fig.set\_facecolor('floralwhite')  
 fig.set\_figwidth(6)  
 fig.set\_figheight(3)  
 #Вывод гистограммы  
 #plt.show()  
  
 fig, ax = plt.subplots()  
 ax.bar(modified\_alphabet, np\_probability\_modified\_alphabet)  
 ax.set\_facecolor('seashell')  
 fig.set\_facecolor('floralwhite')  
 fig.set\_figwidth(6)  
 fig.set\_figheight(3)  
 #Вывод гистограммы  
 #plt.show()  
  
  
def PortCipher(message, alphabet):  
 if(alphabet == "latinica"):  
 start\_symbol = 65  
 end\_symbol = 91  
 power\_alphabet = end\_symbol - start\_symbol  
 elif(alphabet == "kirillica"):  
 start\_symbol = 1040  
 end\_symbol = 1072  
 power\_alphabet = end\_symbol - start\_symbol  
  
 original\_alphabet = np.zeros((power\_alphabet + 1), dtype=str)  
 for letter in range(start\_symbol, end\_symbol, 1):  
 original\_alphabet[letter - start\_symbol] = chr(letter)  
 original\_alphabet[letter - start\_symbol + 1] = " "  
  
 dict\_port = {}  
 for row\_elem in range(len(original\_alphabet)):  
 for col\_elem in range(len(original\_alphabet)):  
 dict\_port[original\_alphabet[row\_elem] + original\_alphabet[col\_elem]] = row\_elem \* (power\_alphabet + 1) + col\_elem  
  
 encrypted\_message = []  
 if(len(message) % 2 != 0):  
 message += " "  
  
 for elem in range(int(len(message) / 2)):  
 encrypted\_message.append(dict\_port.get(message[elem \* 2] + message[elem \* 2 + 1]))  
  
 dict\_port\_inverted = {}  
 for key, values in dict\_port.items():  
 dict\_port\_inverted[values] = dict\_port\_inverted.get(values, []) + [key]  
  
 crypted\_message = ""  
 for elem in encrypted\_message:  
 crypted\_message += str(dict\_port\_inverted.get(elem))[2:len(str(dict\_port\_inverted.get(elem))) - 2]  
 print("Исходное сообщение:", message)  
 print("Зашифрованное сообщение:", encrypted\_message)  
 print("Расшифрованное сообщение:", crypted\_message)  
  
  
main()

**Результат выполнения программы**

