Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЁТ

к лабораторной работе №2

на тему

**ЭЛЕМЕНТЫ КРИПТОГРАФИИ**

Выполнил: студент гр. 253505

Сенько Н. С.

Проверил: ассистент кафедры

информатики

Герчик А. В.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#__RefHeading___Toc1348_1888910669)

[2 Ход работы 4](#__RefHeading___Toc1350_1888910669)

[Заключение 5](#__RefHeading___Toc1352_1888910669)

[Приложение А (обязательное](#__RefHeading___Toc3241_1888910669)) и[сходный код программного продукта 6](#__RefHeading___Toc3243_1888910669)

# 1 ЦЕЛЬ РАБОТЫ

Требуется изучить и реализовать программно классические методы криптографии, такие как шифр Цезаря и шифр Виженера. В процессе работы планируется создание программных инструментов, которые позволят выполнять шифрование и дешифрование текстовых файлов с использованием этих методов. Необходимо разработать алгоритмы для шифрования и расшифровки для обоих методов.

# 2 ХОД РАБОТЫ

Шифр Цезаря – это метод шифрования, при котором каждая буква в открытом тексте заменяется другой буквой, находящейся на фиксированное число позиций дальше в алфавите. Например, при сдвиге на 5 буква «А» превращается в «Е».​

Шифр Виженера – это метод многозначного шифрования, при котором используются различные сдвиги символов в зависимости от ключевого слова. Этот метод является улучшенной версией шифра Цезаря и позволяет значительно усложнить расшифровку без знания ключа. В ходе выполнения лабораторной работы были реализованы функции шифрования и дешифрования текста с использованием шифра Цезаря. Реализованы функции шифрования и дешифрования текста с использованием шифра Виженера. Разработан интерфейс для работы с текстовыми файлами, позволяющий пользователю выбрать метод шифрования, ввести ключ и зашифровать/расшифровать файл.

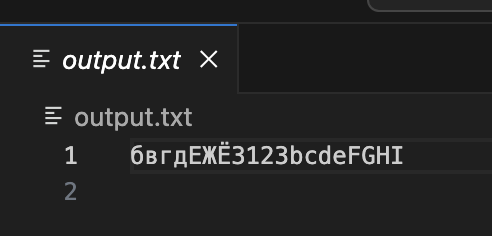


Рисунок 1 – Результат работы программы при шифровании

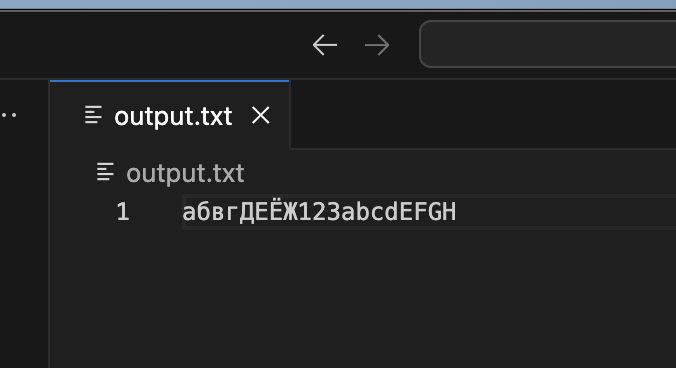


Рисунок 2 – Результат работы программы при дешифрации

Таким образом, в ходе выполнения лабораторной работы было реализовано и протестировано программное средство шифрования и дешифрования текстовых файлов двумя шифрами. В процессе разработки были изучены принципы работы выбранных алгоритмов шифрования, их особенности, преимущества и недостатки.

Программа была протестирована на различных входных данных, что позволило убедиться в её корректности и работоспособности. Кроме того, в ходе выполнения лабораторной работы была проведена отладка кода, исправлены возможные ошибки и оптимизированы некоторые участки программы для повышения её производительности.

# ЗАКЛЮЧЕНИЕ

В ходе работы было создано и протестировано программное приложение для шифрования и дешифрования текстовых файлов с использованием шифров Цезаря и Виженера. Были изучены основные принципы действия этих методов, их характеристики, а также различия в устойчивости к криптоанализу. Шифр Цезаря оказался прост в реализации, но имеет низкую криптостойкость, что делает его подходящим лишь для демонстрационных целей. В свою очередь, шифр Виженера предлагает более сложное шифрование благодаря использованию ключевого слова, что затрудняет его взлом с помощью частотного анализа. В итоге, проведённая работа углубила знания в области симметричного шифрования и защиты информации, а также предоставила возможность практического применения навыков программирования для реализации криптографических методов.

# Приложение А

# (обязательное)

# Исходный код программного продукта

fun main() {

val inp = java.io.File("input.txt")

val out = java.io.File("output.txt")

val ruC = ('А'..'Я').toList()

val ru = ('а'..'я').toList()

val enC = ('A'..'Z').toList()

val en = ('a'..'z').toList()

println("Ключ = (- чтобы расшифровать)")

val key = readlnOrNull()?.toInt() ?: 0

inp.readLines().forEach { line ->

val ans = StringBuilder()

line.forEach { smb ->

if(smb !in (en + enC + ru + ruC)) {

ans.append(smb)

} else {

val isCap = smb in (enC + ruC)

val isEn = smb in (enC + en)

val c = if (isEn) {

if (isCap) {

enC[(smb.code - enC[0].code + key + 10 \* en.size) % en.size]

} else {

en[(smb.code - en[0].code + key + 10 \* en.size) % en.size]

}

} else {

if (isCap) {

ruC[(smb.code - ruC[0].code + key + 10 \* ru.size) % ru.size]

} else {

ru[(smb.code - ru[0].code + key + 10 \* ru.size) % ru.size]

}

}

ans.append(c)

}

}

out.writeText(ans.toString())

}

}

fun encrypt(text: String, keyEn: String, keyRu: String): String {

val fullKeyEn = keyEn.repeat((text.length / keyEn.length) + 1).substring(0, text.length)

val fullKeyRu = keyRu.repeat((text.length / keyRu.length) + 1).substring(0, text.length)

return text.mapIndexed { index, char ->

if (char.lowercaseChar() in (('a'..'z').toList() + ('а'..'я').toList())) {

if (char.lowercaseChar() in 'a'..'z') {

val offset = (if (char in 'A'..'Z') 'A' else 'a').code

val keyChar = fullKeyEn[index].code

((char.code + keyChar - 2 \* offset) % 26 + offset).toChar()

} else {

val offset = (if (char in 'А'..'Я') 'А' else 'а').code

val keyChar = fullKeyRu[index].code

((char.code + keyChar - 2 \* offset) % 33 + offset).toChar()

}

} else {

char

}

}.joinToString("")

}

fun decrypt(text: String, keyEn: String, keyRu: String): String {

val fullKeyEn = keyEn.repeat((text.length / keyEn.length) + 1).substring(0, text.length)

val fullKeyRu = keyRu.repeat((text.length / keyRu.length) + 1).substring(0, text.length)

return text.mapIndexed { index, char ->

if (char.lowercaseChar() in (('a'..'z').toList() + ('а'..'я').toList())) {

if (char.lowercaseChar() in 'a'..'z') {

val offset = (if (char in 'A'..'Z') 'A' else 'a').code

val keyChar = fullKeyEn[index].code

((char.code - keyChar + 52) % 26 + offset).toChar()

} else {

val offset = (if (char in 'А'..'Я') 'А' else 'а').code

val keyChar = fullKeyRu[index].code

((char.code - keyChar + 66) % 33 + offset).toChar()

}

} else {

char

}

}.joinToString("")

}

fun main() {

val inp = java.io.File("input1.txt")

val out = java.io.File("output1.txt")

val i = inp.readLines().toMutableList()

val keyEn = i[0]

val keyRu = i[1]

i.removeFirst()

i.removeFirst()

val text = i.joinToString()

val encrypted = encrypt(text, keyEn, keyRu)

val decrypted = decrypt(text, keyEn, keyRu)

out.writeText("Encrypted: $encrypted\nDecrypted: $decrypted")

}