МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



ЗВІТ

про виконання лабораторної роботи № 1

«Симетричні методи шифрування інформації»

з дисципліни

«Технології захисту інформації»

Виконала:

студентка групи ІТ-31

Щербак Л. В. Прийняв викладач:

Досин Д.Г.

Львів 2022

**Мета роботи:** Навчитися опрацьовувати (шифрувати та дешифрувати) файли на основі методів симетричного шифрування.

**Завдання роботи:**

1. Написати програму на мові С++ (чи іншій за згодою викладача) яка

виконує криптографічні перетворення (шифрування та дешифрування) над

файлами за одним з методів симетричного шифрування відповідно до

заданого варіанту приведеного в таблиці.

Перестановочний шифр із ключем для англ. мови. Результат

представити в бітовому представленні (в вісімковій системі числення)

2. Здійснити компіляцію програми.

3. Результати подати у вигляді текстів програми, відповідних пояснень

та скріншотів.

4. Додаткове завдання. Провести порівняння розподілу ймовірностей

наявних символів файлу до та після шифрування. Результати ймовірностей

представити у табличній та графічній формах для кожного символу

5. Оформити звіт згідно вимог.

**Текст програми реалізації:**

package lab1;

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.HashMap;

import java.util.Map;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class Encription {

static String key = "binary";

static String text;// = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

static char[] sorted\_key;

static int[][] a\_key;

static int x;

static int y;

static char[][] arr1;

static char[][] arr2;

static Map<Integer, Integer> dict\_key = new HashMap<Integer, Integer>();

//get index of symbol in array

public static int getIndex(final char c, char[] arr) {

final int length = arr.length;

for(int i = 0; i < length; i++) {

if (arr[i] == c) {

return i;

}

}

return -1;

}

// read text(english alphabet) from text file

public static void readTextFile() {

final Path file = Paths.get("./lab1.txt");

try {

text = Files.readString(file);

} catch (final Exception e) {

System.out.println(e);

}

}

// save table (as object) with encrypted text to binary file

public static void saveFile(char[][] arr) {

try {

ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream("lab1.encrypted"));

outputStream.writeObject(arr);

} catch (IOException e) {

throw new RuntimeException(e);

}

}

// read table (as object) with encrypted text from binary file

public static void readEncryptedFile() {

try {

ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream("lab1.encrypted"));

arr2 = (char[][])inputStream.readObject();

} catch (IOException e) {

throw new RuntimeException(e);

} catch (ClassNotFoundException e) {

throw new RuntimeException(e);

}

}

//print array as table

public static void print(final char[][] arr) {

for(int i=0; i < y; i++) {

for(int j=0; j < x; j++) {

System.out.print(arr[i][j]);

}

System.out.println();

}

System.out.println();

}

//prepare and fill table with key, indexes and text

public static void fillTable() {

for(int j=0; j < x; j++) {

arr1[0][j] = key.toCharArray()[j];

arr1[1][j] = (dict\_key.get(j)).toString().toCharArray()[0];

}

int k = 0;

for(int i=2; i < y; i++) {

for(int j=0; j < x; j++) {

if (k < text.toCharArray().length) {

arr1[i][j] = text.toCharArray()[k];

k++;

}

else {

arr1[i][j] = ' ';

}

}

}

}

//encrypt table

public static void encrypt() {

for(int j=0; j < x; j++) {

arr2[0][j] = sorted\_key[j];

arr2[1][j] = String.valueOf(j).toCharArray()[0];

}

for(int i=2; i < y; i++) {

for(int j=0; j < x; j++) {

arr2[i][j] = arr1[i][(getIndex(sorted\_key[j], key.toCharArray()))];

}

}

}

//decrypt table

public static void decrypt() {

for(int j=0; j < x; j++) {

arr1[0][j] = key.toCharArray()[j];

arr1[1][j] = (dict\_key.get(j)).toString().toCharArray()[0];

}

for(int i=2; i < y; i++) {

for(int j=0; j < x; j++) {

arr1[i][j] = arr2[i][dict\_key.get(j)];

}

}

}

public static void main(String[] args) {

//read text from text file

readTextFile();

// sort symbols in array(key) by alphabet order

//sorted\_key = key.toCharArray();

//Arrays.sort(sorted\_key);

sorted\_key =

Stream.of(key.split(""))

.sorted()

.collect(Collectors.joining()).toCharArray();

x = sorted\_key.length;//column

y = (int) Math.ceil((double)text.toCharArray().length/sorted\_key.length) + 2;//row

//decrypted table

arr1 = new char[y][x];

//encrypted table

arr2 = new char[y][x];

//prepare dictionary with indexes for key

for(int i=0; i < sorted\_key.length; i++) {

dict\_key.put(getIndex(sorted\_key[i], key.toCharArray()), i);

}

System.out.println("Key: "+key);

System.out.println("String: "+text);

System.out.println();

System.out.println("Table (Key+String) before encryption: ");

fillTable();

print(arr1);

//encryption

System.out.println("Table (Key+String) after encryption: ");

encrypt();

print(arr2);

saveFile(arr2);

System.out.println("String after encryption (converted to Octal): "+"");

for(int i=0; i < x; i++) {

for(int j=2; j < y; j++) {

//System.out.print(Integer.toBinaryString((int)result\_text[i][dict\_key.get(j)]));

//System.out.print(Integer.toOctalString((int)result\_text[i][dict\_key.get(j)]));

//System.out.print(arr2[j][i]);

System.out.print(Integer.toOctalString((int) arr2[j][i]));

}

}

System.out.println();

System.out.println();

//decryption

System.out.println("Table (Key+String) after decryption: "+"");

readEncryptedFile();

decrypt();

print(arr1);

System.out.println("String after decryption: ");

for(int i=2; i < y; i++) {

for(int j=0; j < x; j++) {

System.out.print(arr1[i][j]);

}

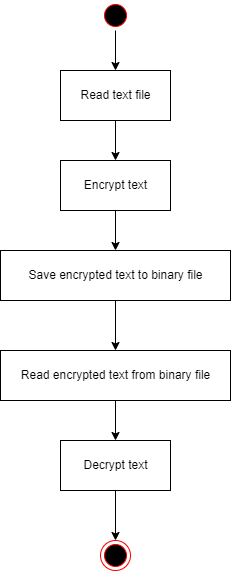
}

System.out.println();

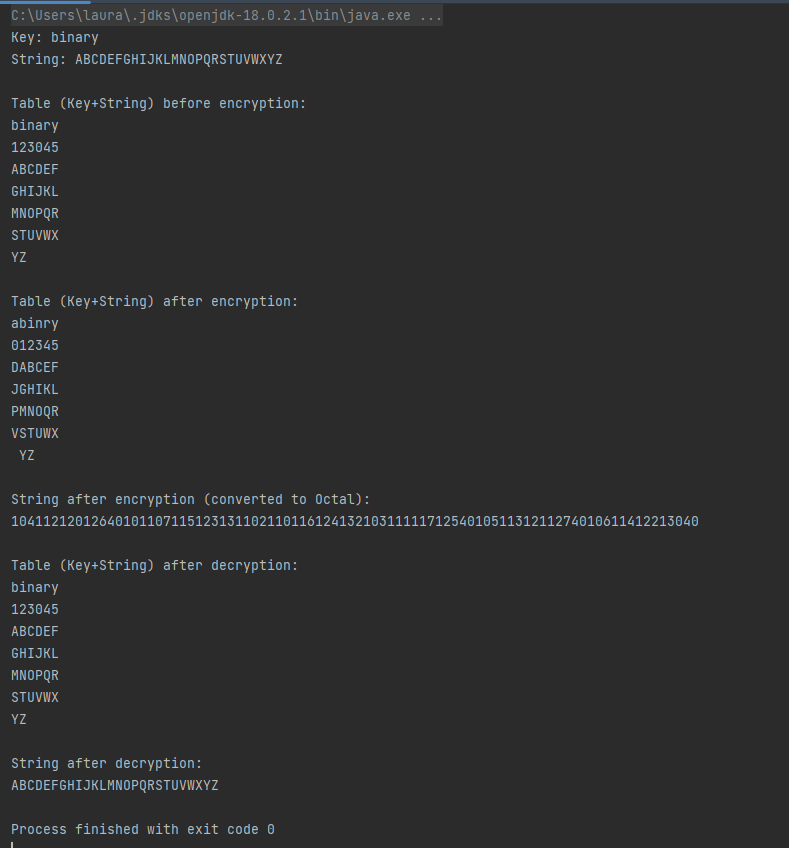
}

}

**Блок-схема:**



**Результати роботи програми:**



**Відповіді на контрольні питання:**

1. Особливості алгоритмів симетричного та асиметричного шифрувань.

Ці два види шифрування відрізняються кількістю ключів і рівнем стійкості до злому. Якщо для кодування і розкодування використовується один ключ, то це симетричне шифрування. Асиметричне шифрування передбачає використання одного ключа для кожного алгоритму. Шифрується відкритим ключем певний код, який являє собою певне послання. Ключ відомий обом сторонам, він передається по незахищеному каналу, може бути перехоплений. Найважливішим завданням збереження інформації є захист ключа від перехоплення. Закритий використовується для розшифрування. Відомий тільки одній стороні. Не може бути перехоплений, так як весь час знаходиться в одного співрозмовника. Мета шифрування визначає метод збереження конфіденційності. Одним з перших було симетричне, асиметричну шифрування винайдено пізніше для забезпечення більшої надійності.

2. Поняття блокового шифрування.

Блоковий шифр дозволяє шифрування тільки одного блоку даних встановленої довжини. Для роботи з блоками різних довжин, дані спочатку потрібно розбити на окремі блоки встановленої даним шифром довжини. Зазвичай, останній блок треба доповнити до відповідної довжини підхожим доповненням. Режими дій описують процес шифрування кожного з цих блоків і звичайно використовують рандомізацію основану на додатковому значені на вході, відомим як ініціалізаційний вектор, з ціллю зробити це безпечно.

3. Шифрування в системі Цезаря.

Алгоритм шифрування Цезаря полягає заміні кожного символу вхідного повідомлення на символ, який знаходиться на деякій константній відстані з правої чи лівої сторони. Відстань при цьому називають – ключем.

Наприклад для ключа 5 отримаємо послідовність:

Алфавіт:

А Б В Г Ґ Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я

Шифр:

Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я А Б В Г Ґ

Тобто А замінюємо на Д, Б на Е, і т. д.

4. Шифрування з допомогою перестановочного шифру.

Криптоаналіз перестановочного шифра виконується за два етапи:

1.Визначається довжина ключа, після чого шифротекст записується у вигляді стовпчиків від 1 до n, де n — довжина ключа.

2. Виконуються перестановки шляхом перебору значень. Для ключа довжиною n необхідно виконати n! перестановок. Це ефективно при n<12.

Якщо ж n>12, доцільно використовувати розміщення перших 2-3 елементів перестановки. Така методика дозволяє відновити ключі n<10^7. Це означає, що не є можливим створити стійкий шифр застосовуючи тільки перестановки.

**Висновки:** Зробила перестановочний шифр із ключем для англ. мови. Результат представила в бітовому представленні (в вісімковій системі числення)