**УКРАЇНИ МІНІСТЕРСТВО ОСВІТИ І НАУКИ**

**СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ**

**ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ**

**КАФЕДРА КОМП’ЮТЕРНИХ НАУК**

**ПРАКТИЧНА РОБОТА № 7**

з дисципліни "Захист інформації"

Тема: **"Шифри cкладної заміни. Шифр Віженера"**

Варіант № 14

**Перевірив Коробченко О. В.**

**Студентки групи Індн – 31с Пархомчук А. К.**

Суми – 2017

**Мета**

Виробити вміння та навички шифрування інформації за допомогою шифру Віженера.

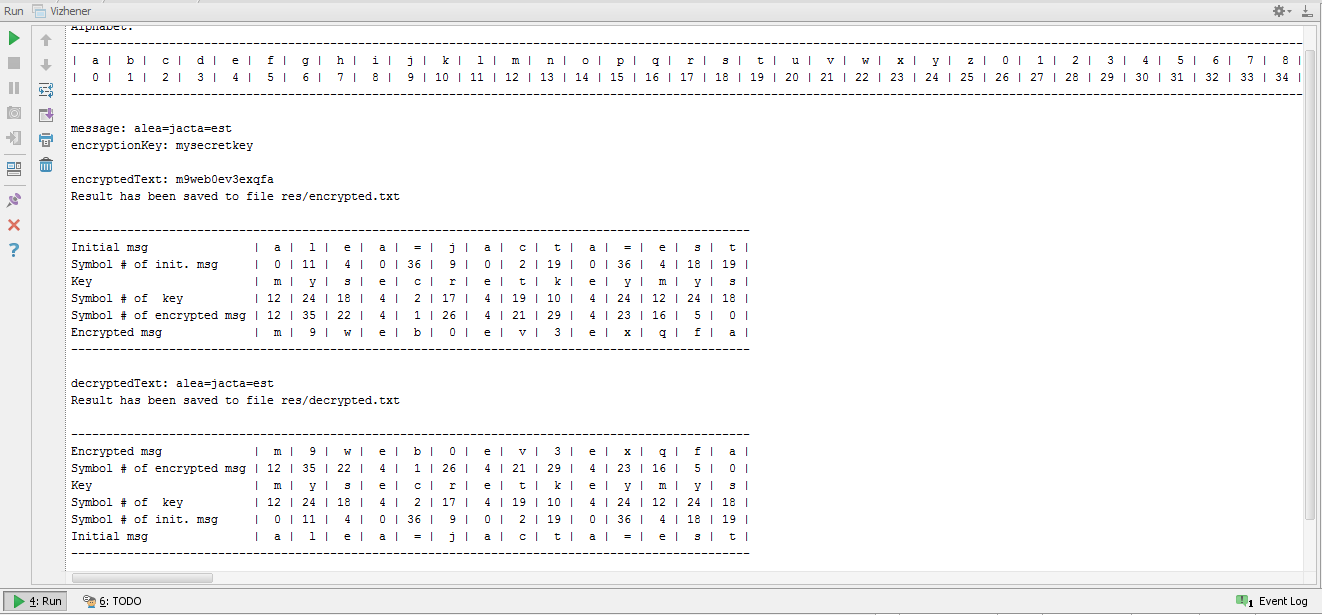
**Завдання**

|  |  |  |
| --- | --- | --- |
| **№**  **в-та** | **А1** | **Тип шифрування** |
| 14 | Малі букви латинського алфавіту, спецсимвол (=) і цифри | За допомогою формули |

**Вихідні дані, які були застосовані для виконання поставленого завдання**

* Алфавіт що містить наступні символи: abcdefghijklmnopqrstuvwxyz0123456789=
* Ключ – mysecretkey
* Текст – alea=jacta=est

**Скриншоти роботи програми**



**Вихідний текст програми**

***Vizhener.java***

import java.io.\*;

public class Vizhener {

public static final String FILE\_IN = "res/in.txt";

public static final String FILE\_ENCRYPTED = "res/encrypted.txt";

public static final String FILE\_DECRYPTED = "res/decrypted.txt";

public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz0123456789=";

public CryptoResult encrypt(String message, String key) {

StringBuffer code = new StringBuffer();

CryptoResult cr = new CryptoResult(message.length(), CryptoResult.ENCRYPT);

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

char miChar = message.charAt(i);

int miIndex = ALPHABET.indexOf(miChar);

cr.setOpenTextCharacter(i, miChar);

cr.setOpenTextIndex(i, miIndex);

char kiChar = key.charAt(i % key.length());

int kiIndex = ALPHABET.indexOf(kiChar);

cr.setKeyTextCharacter(i, kiChar);

cr.setKeyIndex(i, kiIndex);

int cIndex = (miIndex + kiIndex) % ALPHABET.length();

char c = ALPHABET.charAt(cIndex);

cr.setEncryptedTextCharacter(i, c);

cr.setEncryptedTextIndex(i, cIndex);

// System.out.printf("\ni:%d M=%c[%d] K=%c[%d] => (%d) mod %d = %c(%d)\n", i, miChar, miIndex, kiChar, kiIndex, (miIndex+ kiIndex), ALPHABET.length(), c, cIndex);

code.append(c);

}

cr.setResult(code.toString());

return cr;

}

public CryptoResult decrypt(String code, String key) {

StringBuffer messageBuffer = new StringBuffer();

CryptoResult cr = new CryptoResult(code.length(), CryptoResult.DECRYPT);

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

int mIndex;

int alphabetSize = ALPHABET.length();

char ciChar = code.charAt(i);

int ciIndex = ALPHABET.indexOf(ciChar);

cr.setEncryptedTextCharacter(i, ciChar);

cr.setEncryptedTextIndex(i, ciIndex);

char kiChar = key.charAt(i % key.length());

int kiIndex = ALPHABET.indexOf(kiChar);

cr.setKeyTextCharacter(i, kiChar);

cr.setKeyIndex(i, kiIndex);

if (ciIndex - ALPHABET.indexOf(key.charAt(i % key.length())) < 0) {

mIndex = alphabetSize - Math.abs((ciIndex - kiIndex % alphabetSize));

} else {

mIndex = (ciIndex - kiIndex) % alphabetSize;

}

char mChar = ALPHABET.charAt(mIndex);

cr.setOpenTextCharacter(i, mChar);

cr.setOpenTextIndex(i, mIndex);

// System.out.printf("\ni:%d C=%c[%d] K=%c[%d] => (%d) mod %d = %c(%d)\n", i, ciChar, ciIndex, kiChar, kiIndex, (ciIndex - kiIndex), ALPHABET.length(), m, mIndex);

messageBuffer.append(mChar);

}

cr.setResult(messageBuffer.toString());

return cr;

}

public String readFile(String fileName) {

try {

BufferedReader br = new BufferedReader(new FileReader(fileName));

String buffer = br.readLine();

br.close();

return buffer;

} catch (IOException ex) {

System.out.println(ex.toString());

return "";

}

}

public void writeFile(String text, String fileName) {

try {

PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(fileName)));

pw.print(text);

pw.close();

} catch (IOException ex) {

System.out.println(ex.toString());

}

}

public static void main(String[] args) {

Vizhener v = new Vizhener();

String encryptionKey = "mysecretkey";

v.outputAlphabet();

// System.out.println("Initial message has been read from file " + FILE\_IN);

// String message = v.readFile(FILE\_IN);

String message = "alea=jacta=est";

System.out.println("message: " + message);

System.out.println("encryptionKey: " + encryptionKey);

CryptoResult encryptedResult = v.encrypt(message, encryptionKey);

String encryptedText = encryptedResult.getResult();

System.out.println("\nencryptedText: " + encryptedText);

v.writeFile(encryptedText, FILE\_ENCRYPTED);

System.out.println("Result has been saved to file " + FILE\_ENCRYPTED);

System.out.println(encryptedResult);

CryptoResult decryptedResult = v.decrypt(encryptedText, encryptionKey);

String decryptedText = decryptedResult.getResult();

System.out.println("decryptedText: " + decryptedText);

v.writeFile(decryptedText, FILE\_DECRYPTED);

System.out.println("Result has been saved to file " + FILE\_DECRYPTED);

System.out.println(decryptedResult);

}

private void outputAlphabet() {

System.out.println("Alphabet: ");

System.out.print("------------------------------------------------------------------------------------------");

System.out.println("------------------------------------------------------------------------------------------------");

System.out.print("|");

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

System.out.printf("%3c |", ALPHABET.charAt(i));

}

System.out.println();

System.out.print("|");

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

System.out.printf("%3d |", i);

}

System.out.print("\n------------------------------------------------------------------------------------------");

System.out.println("------------------------------------------------------------------------------------------------\n");

}

}

***CryptoResult.java***

public class CryptoResult {

public static final int ENCRYPT = 0;

public static final int DECRYPT = 1;

private char[] openText;

private int[] openTextIndexes;

private char[] keyText;

private int[] keyIndexes;

private char[] encryptedText;

private int[] encryptedTextIndexes;

private int cryptoType;

private String result;

public CryptoResult(int size, int cryptoType) {

this.cryptoType = cryptoType;

openText = new char[size];

openTextIndexes = new int[size];

keyText = new char[size];

keyIndexes = new int[size];

encryptedText = new char[size];

encryptedTextIndexes = new int[size];

}

public void setOpenTextCharacter(int position, char c) {

this.openText[position] = c;

}

public void setOpenTextIndex(int position, int index) {

this.openTextIndexes[position] = index;

}

public void setKeyTextCharacter(int position, char c) {

this.keyText[position] = c;

}

public void setKeyIndex(int position, int index) {

this.keyIndexes[position] = index;

}

public void setEncryptedTextCharacter(int position, char c) {

this.encryptedText[position] = c;

}

public void setEncryptedTextIndex(int position, int index) {

this.encryptedTextIndexes[position] = index;

}

public String getResult() {

return result;

}

public void setResult(String result) {

this.result = result;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

sb.append("\n-------------------------------------------------------------------------------------------------\n");

if (cryptoType == ENCRYPT) {

appendCharacters(sb, "Initial msg |", openText);

appendIntegerNumbers(sb, "\nSymbol # of init. msg |", openTextIndexes);

} else {

appendCharacters(sb, "Encrypted msg |", encryptedText);

appendIntegerNumbers(sb, "\nSymbol # of encrypted msg |", encryptedTextIndexes);

}

appendCharacters(sb, "\nKey |", keyText);

appendIntegerNumbers(sb, "\nSymbol # of key |", keyIndexes);

if (cryptoType == ENCRYPT) {

appendIntegerNumbers(sb, "\nSymbol # of encrypted msg |", encryptedTextIndexes);

appendCharacters(sb, "\nEncrypted msg |", encryptedText);

} else {

appendIntegerNumbers(sb, "\nSymbol # of init. msg |", openTextIndexes);

appendCharacters(sb, "\nInitial msg |", openText);

}

sb.append("\n-------------------------------------------------------------------------------------------------\n");

return sb.toString();

}

private void appendCharacters(StringBuilder sb, String title, char[] chars) {

sb.append(title);

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

sb.append(String.format("%3c |", chars[i]));

}

}

private void appendIntegerNumbers(StringBuilder sb, String title, int[] numbers) {

sb.append(title);

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

sb.append(String.format("%3d |", numbers[i]));

}

}

}

**Результати роботи програми**

Alphabet:

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | = |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

message: alea=jacta=est

encryptionKey: mysecretkey

encryptedText: m9web0ev3exqfa

Result has been saved to file res/encrypted.txt

-------------------------------------------------------------------------------------------------

Initial msg | a | l | e | a | = | j | a | c | t | a | = | e | s | t |

Symbol # of init. msg | 0 | 11 | 4 | 0 | 36 | 9 | 0 | 2 | 19 | 0 | 36 | 4 | 18 | 19 |

Key | m | y | s | e | c | r | e | t | k | e | y | m | y | s |

Symbol # of key | 12 | 24 | 18 | 4 | 2 | 17 | 4 | 19 | 10 | 4 | 24 | 12 | 24 | 18 |

Symbol # of encrypted msg | 12 | 35 | 22 | 4 | 1 | 26 | 4 | 21 | 29 | 4 | 23 | 16 | 5 | 0 |

Encrypted msg | m | 9 | w | e | b | 0 | e | v | 3 | e | x | q | f | a |

-------------------------------------------------------------------------------------------------

decryptedText: alea=jacta=est

Result has been saved to file res/decrypted.txt

-------------------------------------------------------------------------------------------------

Encrypted msg | m | 9 | w | e | b | 0 | e | v | 3 | e | x | q | f | a |

Symbol # of encrypted msg | 12 | 35 | 22 | 4 | 1 | 26 | 4 | 21 | 29 | 4 | 23 | 16 | 5 | 0 |

Key | m | y | s | e | c | r | e | t | k | e | y | m | y | s |

Symbol # of key | 12 | 24 | 18 | 4 | 2 | 17 | 4 | 19 | 10 | 4 | 24 | 12 | 24 | 18 |

Symbol # of init. msg | 0 | 11 | 4 | 0 | 36 | 9 | 0 | 2 | 19 | 0 | 36 | 4 | 18 | 19 |

Initial msg | a | l | e | a | = | j | a | c | t | a | = | e | s | t |

-------------------------------------------------------------------------------------------------

**Висновки**

В ході виконання роботи отримані навички шифрування інформації за допомогою шифру Віженера. При виконанні роботи за допомогою формули Віженера для алфавіту: abcdefghijklmnopqrstuvwxyz0123456789= зашифровано вихідний текст: alea=jacta=est за допомогою ключа mysecretkey. Отримано зашифрований текст: m9web0ev3exqfa. Проведено дешифрування зашифрованого тексту за допомогою ключа mysecretkey. При дешифруванні отримано вихідний текст: alea=jacta=est.