Міністерство освіти та науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та

Комп’ютерних технологій

**Звіт**

Про виконання лабораторної роботи №3

“Програмування класичних криптоалгоритмів”

Виконав:

Студент групи ФеІ-44

Сапанюк М.І.

Перевірив:

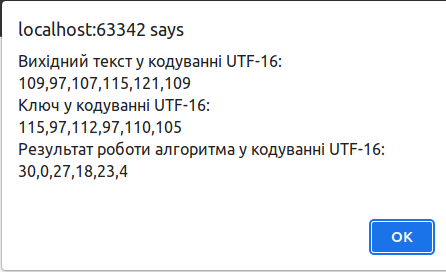
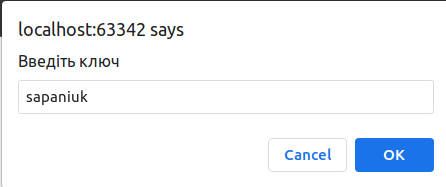
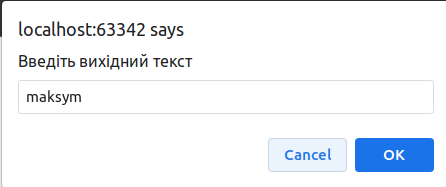
Гура В. Т.

Львів 2022

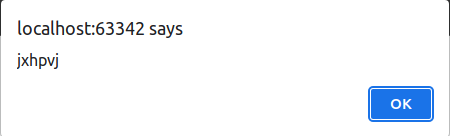
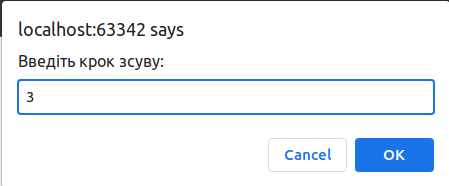
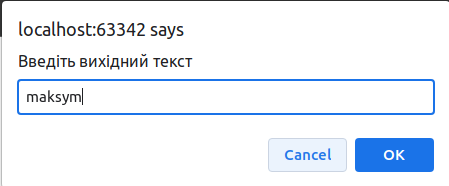
**Мета роботи:** запрограмувати класичні криптоалгоритми

**Порядок виконання роботи:**

1. Шифр Вернама

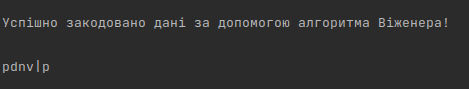
function getRandomInt(min, max) {  
 return ***Math***.floor(***Math***.random() \* (max - min)) + min;  
}  
  
var ***input***, ***output***, ***key***;  
  
var ***inp*** = [], ***k*** = [];  
  
input = prompt("Введіть вихідний текст");  
  
key = prompt("Введіть ключ");  
  
if ((***key***.length) < (***input***.length)) {  
 alert("Ключ коротше повідомлення це небезпечно. Скопіюйте новий згенерований ключ із консолі браузера.");  
 ***key*** = "";  
 for (var ***i*** = 0; ***i*** < ***input***.length; ***i***++) {  
 ***key*** += String.fromCharCode(getRandomInt(0, 66535));  
 }  
 ***console***.log("Скопіюйте новий ключ:");  
 ***console***.log(***key***);  
  
}  
  
output = [];  
for (***i*** = 0; ***i*** < ***input***.length; ***i***++) {  
 ***inp***.push(***input***.charCodeAt(***i***));  
 ***k***.push(***key***.charCodeAt(***i***));  
 ***output***.push(***inp***[***i***] ^ ***k***[***i***]);  
}  
  
alert(`Вихідний текст у кодуванні UTF-16: \n${***inp***}  
Ключ у кодуванні UTF-16: \n${***k***}  
Результат роботи алгоритма у кодуванні UTF-16: \n${***output***}`);  


1. Шифр Цезаря

TextToWork = prompt('Введіть вихідний текст')  
UserStep = Number(prompt('Введіть крок зсуву:'))  
  
var ***OtherSymbols*** = [' ', ',', '.', ':', ';', '!', '?', '-', '\_', '=', '+', '(', ')', '[', ']', '@', '`', "'", '"', '<', '>', '|', '/', '%', '$', '^', '&', '\*', '~'];  
var ***Numbers*** = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'];  
var ***EngAlfUp*** = ['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'];  
var ***EngAlfLower*** = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'm', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'];  
var ***EngAlfUpEncrypt*** = ***Array***(26);  
var ***EngAlfLowerEncrypt*** = ***Array***(26);  
var ***NumbersEncrypt*** = ***Array***(10);  
  
initEncrypt();  
  
function initEncrypt() {  
 ***NumbersEncrypt*** = CezarEncrypt(UserStep, ***Numbers***);  
 ***EngAlfUpEncrypt*** = CezarEncrypt(UserStep, ***EngAlfUp***);  
 ***EngAlfLowerEncrypt*** = CezarEncrypt(UserStep, ***EngAlfLower***);  
}  
  
function CezarEncrypt(step, arr) {  
 var copy = arr.slice();  
 for (i = 0; i < step; i += 1) {  
 copy.unshift(copy.pop())  
 }  
 return copy;  
}  
  
function contains(symb, arr) {  
 var letter = symb;  
 pos = 0;  
 for (var i = 0; i < arr.length; i++) {  
 if (letter === arr[i]) {  
 pos = i;  
 return true;  
 }  
 }  
}  
  
function encrypt(text) {  
 var result = '';  
 for (var i = 0; i <= text.length; i++) {  
 var symbol = text[i];  
 if (contains(symbol, ***OtherSymbols***)) {  
 result += symbol;  
 }  
 if (contains(symbol, ***Numbers***)) {  
 symbol = ***NumbersEncrypt***[pos];  
 result += symbol;  
 }  
 if (contains(symbol, ***EngAlfUp***)) {  
 symbol = ***EngAlfUpEncrypt***[pos];  
 result += symbol;  
 }  
 if ((contains(symbol, ***EngAlfLower***))) {  
 symbol = ***EngAlfLowerEncrypt***[pos];  
 result += symbol;  
 }  
 }  
 return result;  
}  
  
alert(encrypt(TextToWork))  


1. Шифр Віженера

const Wegener = (text, k) => {  
 const chars = text.split('');  
 const encodedData = chars  
 .map((char) => String.fromCharCode(char.charCodeAt() + k))  
 .join('');  
 ***console***.log(  
 `\nУспішно закодовано дані за допомогою алгоритма Віженера!\n`  
 );  
 ***console***.log(encodedData)  
}  
Wegener('maksym', 3)



1. Шифр Хілла

function hill(string, key) {  
 'use strict';  
 if (!string || !key && !string.length > 0 || !key.length > 0) return;  
 var s = this, w = '',  
 a = ['', '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', '.', ',', '?', '%', '/', ':', '-', '&', '7', '1'],  
 u = string.split(''), k = key.split('');  
 ***Array***.prototype.chunk = function (groupsize) {  
 var sets = [], chunks, i = 0;  
 chunks = this.length / groupsize;  
 while (i < chunks) {  
 sets[i] = this.splice(0, groupsize);  
 if (sets[i].length < groupsize) {  
 var diff = groupsize - sets[i].length;  
 for (let j = 0; j < diff; j++) {  
 sets[i].push(0);  
 }  
 }  
 i++;  
 }  
 return sets;  
 };  
 ***Array***.prototype.arrayToMatrix = function (elementsPerSubArray) {  
 var matrix = [], i, k;  
 for (i = 0, k = -1; i < this.length; i++) {  
 if (i % elementsPerSubArray === 0) {  
 k++;  
 matrix[k] = [];  
 }  
 matrix[k].push(this[i]);  
 }  
 return matrix;  
 }  
 s.multiplyMatrix = (a, b) => {  
 var aNumRows = a.length, aNumCols = a[0].length, bNumRows = b.length, bNumCols = b[0].length,  
 m = new ***Array***(aNumRows);  
 for (var r = 0; r < aNumRows; ++r) {  
 m[r] = new ***Array***(bNumCols);  
 for (var c = 0; c < bNumCols; ++c) {  
 m[r][c] = 0;  
 for (var i = 0; i < aNumCols; ++i) {  
 m[r][c] += a[r][i] \* b[i][c];  
 }  
 }  
 }  
 return m;  
 }  
 s.determinantMatrix = (matrix) => {  
 var N = matrix.length, B = [], denom = 1, exchanges = 0;  
 for (var i = 0; i < N; ++i) {  
 B[i] = [];  
 for (var j = 0; j < N; ++j) B[i][j] = matrix[i][j];  
 }  
 for (var i = 0; i < N - 1; ++i) {  
 var maxN = i, maxValue = ***Math***.abs(B[i][i]);  
 for (var j = i + 1; j < N; ++j) {  
 var value = ***Math***.abs(B[j][i]);  
 if (value > maxValue) {  
 maxN = j;  
 maxValue = value;  
 }  
 }  
 if (maxN > i) {  
 var temp = B[i];  
 B[i] = B[maxN];  
 B[maxN] = temp;  
 ++exchanges;  
 } else {  
 if (maxValue == 0) return maxValue;  
 }  
 var value1 = B[i][i];  
 for (var j = i + 1; j < N; ++j) {  
 var value2 = B[j][i];  
 B[j][i] = 0;  
 for (var k = i + 1; k < N; ++k) B[j][k] = (B[j][k] \* value1 - B[i][k] \* value2) / denom;  
 }  
 denom = value1;  
 }  
 if (exchanges % 2) return -B[N - 1][N - 1]; else return B[N - 1][N - 1];  
 }  
 for (let i = 0; i < u.length; i++) {  
 for (let j = 0; j < a.length; j++) {  
 if (u[i] === a[j]) {  
 u[i] = j;  
 }  
 }  
 }  
 for (let i = 0; i < k.length; i++) {  
 for (let j = 0; j < a.length; j++) {  
 if (k[i] === a[j]) {  
 k[i] = j;  
 }  
 }  
  
 }  
 var keysqrt = ***Math***.sqrt(k.length);  
 if (!k.length > 1 && !(keysqrt % 1) === 0) return;  
 k = k.arrayToMatrix(keysqrt);  
 u = u.chunk(k.length)  
 s.encrypt = () => {  
 for (let i = 0; i < u.length; i++) {  
 let newMatrix = s.multiplyMatrix([u[i]], k)[0];  
 for (let j = 0; j < newMatrix.length; j++) {  
 for (let q = 0; q < a.length; q++) {  
 if (newMatrix[j] % a.length === q) {  
 w += a[q];  
 }  
 }  
 }  
 }  
 return w;  
 }  
 s.decrypt = () => {  
 var matrixR = [], matrixA = k, m = a.length;  
 var outerParent = this;  
 var test = new function () {  
 var innerParent = this;  
 var Calculate3312\_result = {};  
 this.Calculate3312 = function (\_\_\_inp\_\_\_) {  
 function \_\_impl\_\_(\_\_inp\_\_) {  
 var progressControl = outerParent.progressControl;  
 var a = typeof (\_\_inp\_\_["a"]) == "undefined" ? 3 : \_\_inp\_\_["a"];  
 var m = typeof (\_\_inp\_\_["m"]) == "undefined" ? 26 : \_\_inp\_\_["m"];  
 var errors = {"e": "Обратного элемента не существует"}  
 Calculate3312\_result.errors = errors  
 var b = {  
 "SetValue": function (v) {  
 Calculate3312\_result["b"] = v;  
 }  
 };  
 var ta = a;  
 if (ta > m) ta = ta % m;  
 if (ta < 0) ta = ta + m \* (***Math***.floor(***Math***.abs(ta) / m) + 1);  
  
 var result = test.Calculate3299({"first": ta, "second": m});  
  
 if (result.res != 1) throw {"source": "a", "message": errors["e"]}; else {  
 var tx = (result.coef2 % m + m) % m;  
 b.SetValue(tx);  
 }  
 };\_\_impl\_\_(\_\_\_inp\_\_\_);  
 return Calculate3312\_result;  
 };  
 var Calculate3299\_result = {};  
 this.Calculate3299 = function (\_\_\_inp\_\_\_) {  
 function \_\_impl\_\_(\_\_inp\_\_) {  
 var progressControl = outerParent.progressControl;  
 var first = typeof (\_\_inp\_\_["first"]) == "undefined" ? 180 : \_\_inp\_\_["first"];  
 var second = typeof (\_\_inp\_\_["second"]) == "undefined" ? 150 : \_\_inp\_\_["second"];  
 var res = {  
 "SetValue": function (v) {  
 Calculate3299\_result["res"] = v;  
 }  
 };  
 var coef1 = {  
 "SetValue": function (v) {  
 Calculate3299\_result["coef1"] = v;  
 }  
 };  
 var coef2 = {  
 "SetValue": function (v) {  
 Calculate3299\_result["coef2"] = v;  
 }  
 };  
 var euklid = {  
 gcd: 1, x: 0, y: 0  
 }  
 function gcd(a, b, holder) {  
 if (a == 0) {  
 holder.x = 0;  
 holder.y = 1;  
 return b;  
 }  
 var d = gcd(b % a, a, holder);  
 var tx = holder.x;  
 holder.x = holder.y - ***Math***.floor(b / a) \* holder.x;  
 holder.y = tx;  
 return d;  
 }  
 euklid.gcd = gcd(first > second ? second : first, first > second ? first : second, euklid);  
 res.SetValue(euklid.gcd);  
 coef1.SetValue(euklid.y);  
 coef2.SetValue(euklid.x);  
 };\_\_impl\_\_(\_\_\_inp\_\_\_);  
 return Calculate3299\_result;  
 };  
 };  
 function MatrixSolver() {  
 this.c\_valid = 0;  
 this.c\_nonsquare = -1;  
 this.c\_singular = -2;  
 this.c\_wrongdimensions = -3;  
 function get(matrixarray, i, j) {  
 var row = matrixarray[i];  
 return row[j];  
 }  
 function columns(matrixarray) {  
 var row = matrixarray[0];  
 return row.length;  
 }  
 function rows(matrixarray) {  
 return matrixarray.length;  
 }  
 function show(matrixarray) {  
 var s = "";  
 for (var i = 0; i < rows(matrixarray); ++i) {  
 for (var j = 0; j < columns(matrixarray); ++j) {  
 var row = matrixarray[i];  
 s += " " + row[j];  
 }  
 s += "\n\r";  
 }  
 }  
 function minor(matrixarray, i, j) {  
 var m = [];  
 for (var k = 0; k < matrixarray.length; ++k) {  
 if (k == i) continue;  
 var row = matrixarray[k];  
 var newrow = [];  
 for (var l = 0; l < row.length; ++l) {  
 if (l == j) continue;  
 newrow.push(row[l]);  
 }  
 m.push(newrow);  
 }  
 return m;  
 }  
 this.calcTransponent = function (matrixarray) {  
 var transponent = [];  
 var cols = columns(matrixarray);  
 for (var i = 0; i < cols; ++i) {  
 var newrow = [];  
 var rowscount = rows(matrixarray);  
 for (var j = 0; j < rowscount; ++j) {  
 newrow[newrow.length] = get(matrixarray, j, i);  
 }  
 transponent[transponent.length] = newrow;  
 }  
 return transponent;  
 }  
 this.calcScalar = function (matrixarray, scalarValue) {  
 var scalar = [];  
 var cols = columns(matrixarray);  
 for (var i = 0; i < cols; ++i) {  
 var newrow = [];  
 var rowscount = rows(matrixarray);  
 for (var j = 0; j < rowscount; ++j) {  
 newrow.push(get(matrixarray, i, j) \* scalarValue);  
 }  
 scalar.push(newrow);  
 }  
 return scalar;  
 }  
 this.calcDeterminant = function (matrixarray) {  
 var columnsA = columns(matrixarray);  
 var rowsA = rows(matrixarray);  
 if (columnsA != rowsA) throw this.c\_nonsquare;  
  
 if (matrixarray.length == 1) return get(matrixarray, 0, 0); else if (matrixarray.length == 2) return get(matrixarray, 0, 0) \* get(matrixarray, 1, 1) - get(matrixarray, 0, 1) \* get(matrixarray, 1, 0); else {  
 var det = 0;  
 var cols = columns(matrixarray);  
 for (var i = 0; i < cols; ++i) {  
 det += ***Math***.pow(-1, i) \* get(matrixarray, 0, i) \* this.calcDeterminant(minor(matrixarray, 0, i));  
 }  
 return det;  
 }  
 }  
 this.calcInverse = function (matrixarray) {  
 var columnsA = columns(matrixarray);  
 var rowsA = rows(matrixarray);  
 if (columnsA != rowsA) throw this.c\_nonsquare;  
 var detA = this.calcDeterminant(matrixarray);  
 if (detA == 0) throw this.c\_singular;  
 var minorsmatrix = [];  
 for (var i = 0; i < rowsA; ++i) {  
 var newrow = [];  
 for (var j = 0; j < columnsA; ++j) {  
 var val = this.calcDeterminant(minor(matrixarray, i, j));  
 val = val \* ***Math***.pow(-1, i + j);  
 newrow.push(val);  
 }  
 minorsmatrix.push(newrow);  
 }  
 var transponentminors = this.calcTransponent(minorsmatrix);  
 var scalarresult = this.calcScalar(transponentminors, 1 / detA);  
 return scalarresult;  
 }  
 function norm(ta, m) {  
 if (ta > m) ta = ta % m;  
 if (ta < 0) ta = ta + m \* (***Math***.floor(***Math***.abs(ta) / m) + 1);  
 return ta;  
 }  
 this.calcScalarMod = function (matrixarray, scalarValue, modulus) {  
 var scalar = [];  
 var cols = columns(matrixarray);  
 for (var i = 0; i < cols; ++i) {  
 var newrow = [];  
 var rowscount = rows(matrixarray);  
 for (var j = 0; j < rowscount; ++j) {  
 newrow.push(norm(get(matrixarray, i, j) \* scalarValue, modulus));  
 }  
 scalar.push(newrow);  
 }  
 return scalar;  
 }  
 this.calcDeterminantMod = function (matrixarray, modulus) {  
 var columnsA = columns(matrixarray);  
 var rowsA = rows(matrixarray);  
 if (columnsA != rowsA) throw this.c\_nonsquare;  
 if (matrixarray.length == 1) return norm(get(matrixarray, 0, 0), modulus); else if (matrixarray.length == 2) {  
 return norm(get(matrixarray, 0, 0) \* get(matrixarray, 1, 1) - get(matrixarray, 0, 1) \* get(matrixarray, 1, 0), modulus);  
 } else {  
 var det = 0;  
 var cols = columns(matrixarray);  
 for (var i = 0; i < cols; ++i) {  
 det += norm(***Math***.pow(-1, i) \* get(matrixarray, 0, i) \* this.calcDeterminantMod(minor(matrixarray, 0, i), modulus), modulus);  
 }  
 return norm(det, modulus);  
 }  
 }  
 this.calcInverseMod = function (matrixarray, modulus) {  
 var columnsA = columns(matrixarray);  
 var rowsA = rows(matrixarray);  
 if (columnsA != rowsA) throw this.c\_nonsquare;  
 var detA = this.calcDeterminantMod(matrixarray, modulus);  
 if (detA == 0) throw this.c\_singular;  
 var minorsmatrix = [];  
 for (var i = 0; i < rowsA; ++i) {  
 var newrow = [];  
 for (var j = 0; j < columnsA; ++j) {  
 var val = this.calcDeterminantMod(minor(matrixarray, i, j), modulus);  
 val = norm(val \* ***Math***.pow(-1, i + j), modulus);  
 newrow.push(val);  
 }  
 minorsmatrix.push(newrow);  
 }  
 var transponentminors = this.calcTransponent(minorsmatrix);  
 var result = test.Calculate3312({"a": detA, "m": modulus});  
 var scalarresult = this.calcScalarMod(transponentminors, result.b, modulus);  
 return scalarresult;  
 }  
 }  
 function MatrixParser() {  
 this.c\_valid = 0;  
 this.c\_invalid = -1;  
 this.c\_empty = -2;  
 this.error\_line = -1;  
 this.rows = -1;  
 this.columns = -1;  
 this.matrix = [];  
 this.parse = function (matrixstring) {  
 var rows = [];  
 for (let j = 0; j < k.length; j++) {  
 rows.push(k[j].join(' '));  
 }  
 for (var i = 0; i < rows.length; ++i) {  
 if (rows[i].length == 0) continue;  
 var cols = rows[i].split(/\s|\t/);  
 var nums = [];  
 for (var j = 0; j < cols.length; ++j) {  
 if (cols[j].length == 0) continue;  
 var num = Number(cols[j].replace(',', '.'));  
 if (isNaN(num)) {  
 this.error\_line = i + 1;  
 throw this.c\_invalid;  
 break;  
 }  
 nums.push(num);  
 }  
 if (this.columns == -1) this.columns = nums.length; else if (this.columns != nums.length) {  
 this.error\_line = i + 1;  
 throw this.c\_invalid;  
 break;  
 }  
 if (nums.length == 0) {  
 throw this.c\_empty;  
 break;  
 }  
 this.matrix.push(nums);  
 }  
 if (this.matrix.length == 0) throw this.c\_empty;  
 this.rows = this.matrix.length;  
 }  
 }  
 var parserA = new MatrixParser();  
 try {  
 parserA.parse(matrixA);  
 } catch (err) {  
 if (err == parserA.c\_empty) throw {  
 "source": "matrixA", "message": messages["Zero"]  
 }; else if (err == parserA.c\_invalid) throw {  
 "source": "matrixA", "message": (messages["Error"] + " " + parserA.error\_line)  
 }; else ***console***.log(err);  
 }  
 var solver = new MatrixSolver();  
 try {  
 var res = solver.calcInverseMod(parserA.matrix, m);  
 matrixR = res;  
 } catch (err) {  
 if (err == solver.c\_nonsquare) throw {  
 "source": "matrixA", "message": messages["NonSquare"]  
 }; else if (err == solver.c\_singular) throw {  
 "source": "matrixA", "message": messages["Singular"]  
 }; else ***console***.log(err);  
 }  
 for (let i = 0; i < u.length; i++) {  
 let newMatrix = s.multiplyMatrix([u[i]], matrixR)[0];  
 for (let j = 0; j < newMatrix.length; j++) {  
 for (let q = 0; q < a.length; q++) {  
 if (newMatrix[j] % a.length === q) {  
 w += a[q];  
 }  
 }  
 }  
 }  
 return w;  
 }  
};  
  
console.log(new hill('maksym', 'skey').encrypt());  
console.log(new hill('%thdvh', 'skey').decrypt());



**Висновок:** на лабораторній роботі я вивчив та запрограмував класичні шифри, результати подав у звіті.