**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет**   
**імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра Інтелектуальних інформаційних систем**

**ЗВІТ**

*З лабораторної роботи №2*

*на тему:* Створення алгоритму шифрування MD5

Спеціальність “122 Комп’ютерні науки та ІТ”

6.122-401.1610108

***Cтудент*** Канатьєв М.Ю.

*\_\_\_\_\_\_\_\_\_\_\_* ***\_\_\_\_\_\_\_\_\_\_\_\_***

*(підпис)*

\_\_\_\_\_\_\_\_\_\_

*(дата)*

***Викладач*** Тогоєв О. Р.

*\_\_\_\_\_\_\_\_\_\_\_* ***\_\_\_\_\_\_\_\_\_\_\_\_***

*(підпис)*

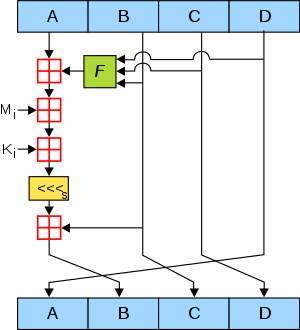
*\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

Миколаїв – 2019

# Теоретичні відомості:

**Алгоритм MD5**

[](https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:MD5_algorithm.svg)

Блок схема роботи алгоритму MD5

**Початковий етап підготовки**

Вхідні дані вирівнюються так, щоб їхній розмір можна було порівняти з 448 по модулю з 512. Спочатку дописують одиничний біт (навіть якщо довжина порівняна з 448), далі необхідна кількість нульових бітів .

Дописування 64-бітного представлення довжини даних по вирівнюванню. Якщо довжина перевищує {\displaystyle 2^{64}-1}, то дописують молодші біти.

**Допоміжні таблиці та функції**

Ініціалізуть 4 змінних розміром по 32 біта:

А = 01 23 45 67;

В = 89 AB CD EF;

С = FE DC BA 98;

D = 76 54 32 10.

Вирівнювані дані розбиваються на блоки по 32 біта, і кожен проходить 4 раунда з 16 операторів. Всі оператори однотипні і мають вигляд: [abcd k s i], визначений як {\displaystyle a=b+((a+Fun(b,c,d)+X[k]+T<i>)<<<s)}, де X - блок даних, а T[1..64] - 64-елементна таблиця, побудована наступним чином: {\displaystyle T[i]=int(4294967296\*|sin(i)|)}, s - циклічний зсув вліво на s біт отриманого 32-бітного аргументу.

В першому раунді Fun F(X, Y, Z) = XY v (not X)Z

В другому раунді Fun G(X, Y, Z) = XZ v (not Z)Y.

В третьому раунді Fun Н(Х, Y, Z) = Х xor Y xor Z.

В четвертому раунді Fun I(Х, Y, Z) = Y xor (X v (not Z)).

**Циклічна процедура обчислення**

Саме обчислення проходить наступним чином:

Зберігаються значення A, B, C і D, що залишились після операцій з попередніми блоками(або їх початкові значення якщо блок перший)

AA = A

BB = B

CC = C

DD = D

**Раунд 1**

/\*[abcd k s i] a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). \*/

[ABCD 0 7 1][DABC 1 12 2][CDAB 2 17 3][BCDA 3 22 4]

[ABCD 4 7 5][DABC 5 12 6][CDAB 6 17 7][BCDA 7 22 8]

[ABCD 8 7 9][DABC 9 12 10][CDAB 10 17 11][BCDA 11 22 12]

[ABCD 12 7 13][DABC 13 12 14][CDAB 14 17 15][BCDA 15 22 16]

**Раунд 2**

/\*[abcd k s i] a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). \*/

[ABCD 1 5 17][DABC 6 9 18][CDAB 11 14 19][BCDA 0 20 20]

[ABCD 5 5 21][DABC 10 9 22][CDAB 15 14 23][BCDA 4 20 24]

[ABCD 9 5 25][DABC 14 9 26][CDAB 3 14 27][BCDA 8 20 28]

[ABCD 13 5 29][DABC 2 9 30][CDAB 7 14 31][BCDA 12 20 32]

**Раунд 3**

/\*[abcd k s i] a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). \*/

[ABCD 5 4 33][DABC 8 11 34][CDAB 11 16 35][BCDA 14 23 36]

[ABCD 1 4 37][DABC 4 11 38][CDAB 7 16 39][BCDA 10 23 40]

[ABCD 13 4 41][DABC 0 11 42][CDAB 3 16 43][BCDA 6 23 44]

[ABCD 9 4 45][DABC 12 11 46][CDAB 15 16 47][BCDA 2 23 48]

**Раунд 4**

/\*[abcd k s i] a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). \*/

[ABCD 0 6 49][DABC 7 10 50][CDAB 14 15 51][BCDA 5 21 52]

[ABCD 12 6 53][DABC 3 10 54][CDAB 10 15 55][BCDA 1 21 56]

[ABCD 8 6 57][DABC 15 10 58][CDAB 6 15 59][BCDA 13 21 60]

[ABCD 4 6 61][DABC 11 10 62][CDAB 2 15 63][BCDA 9 21 64]

**Проміжний результат**

**Виконати наступні операції**

A = AA + A

B = BB + B

C = CC + C

D = DD + D

Після цього перевірити, чи є ще блоки, якщо є, то повторюють циклічну процедуру обчислення для наступного 32-х бітового блоку.

**Результат**

Після обчислення для всіх блоків даних, отримуємо кінцевий хеш у регістрах A B C D. Якщо вивести слова у зворотному порядку DCBA, то отримаємо MD5 хеш.

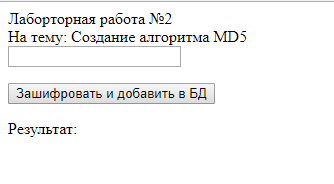
Завдання:

* На будь-які мові програмування відтворити алгоритм шифрування MD5.
* Результат повинен бути записаним в базу даних(на вибір студента).
* Створити репозиторій github з викладеними файлами та звітом.

**Виконання роботи:**

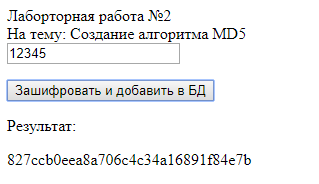
В даній лабораторну роботу було виконано на мові програмування PHP, JavaScript та JQuery.

Спочатку було створено просту HTML форму вводу даних для шифрування.

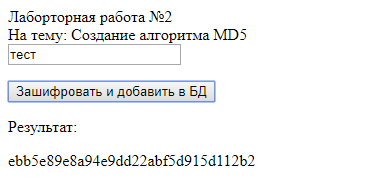


Сторінка має поле ввода та кнопку.

Кнопка шифрує текст та додає його до бази даних



****

****

****

**Лістинг програми:**

**titul.html:**

<html>

<head>

<script type="text/javascript" src="jquery.js">

</script>

<script type="text/javascript">

$( document ).ready(function() {

console.log( "ready!" );

$( "#cryptButton" ).on( "click", function() {

var data = {'string': $('#textField')[0].value};

console.log('-=data=-', data);

$.ajax({

type: "POST",

data: data,

url: 'http://localhost/index.php',

success: function(xhr){

console.log('-=xhr=-', xhr);

$('#result').text(xhr);

},

});

});

});

</script>

</head>

<body>

<div>

Лаборторная работа №2

</br>На тему: Создание алгоритма MD5

</div>

<div>

<form>

<input id="textField">

</input>

</form>

<button id="cryptButton">

Зашифровать и добавить в БД

</button>

</div>

<div>

<p>

Результат:

</p>

<span id="result">

</span>

</div>

</body>

</html>

**Index.php:**

<?php

include './cryptClass.php';

$string = $\_POST['string'];

$result = MD($string);

$link = mysqli\_connect('localhost', 'root', '', 'md5');

if (!$link) {

die('Ошибка соединения: ' . mysql\_error());

} else {

$link->real\_query('INSERT INTO md5hash (hash) VALUES ("'.$result.'")');

}

mysqli\_close($link);

header('Access-Control-Allow-Origin: \*');

header ("Content-Type: text/html; charset=utf-8");

echo $result;

?>

**CryptClass.php**

<?php

function MD($string){

$a = "67452301";

$b = "EFCDAB89";

$c = "98BADCFE";

$d = "10325476";

$words = init($string);

for($i = 0; $i <= count($words)/16-1; $i++){

$A = $a;

$B = $b;

$C = $c;

$D = $d;

FF ($A, $B, $C, $D, $words[0 + ($i \* 16)], 7, "d76aa478");

FF ($D, $A, $B, $C, $words[1 + ($i \* 16)], 12, "e8c7b756");

FF ($C, $D, $A, $B, $words[2 + ($i \* 16)], 17, "242070db");

FF ($B, $C, $D, $A, $words[3 + ($i \* 16)], 22, "c1bdceee");

FF ($A, $B, $C, $D, $words[4 + ($i \* 16)], 7, "f57c0faf");

FF ($D, $A, $B, $C, $words[5 + ($i \* 16)], 12, "4787c62a");

FF ($C, $D, $A, $B, $words[6 + ($i \* 16)], 17, "a8304613");

FF ($B, $C, $D, $A, $words[7 + ($i \* 16)], 22, "fd469501");

FF ($A, $B, $C, $D, $words[8 + ($i \* 16)], 7, "698098d8");

FF ($D, $A, $B, $C, $words[9 + ($i \* 16)], 12, "8b44f7af");

FF ($C, $D, $A, $B, $words[10 + ($i \* 16)], 17, "ffff5bb1");

FF ($B, $C, $D, $A, $words[11 + ($i \* 16)], 22, "895cd7be");

FF ($A, $B, $C, $D, $words[12 + ($i \* 16)], 7, "6b901122");

FF ($D, $A, $B, $C, $words[13 + ($i \* 16)], 12, "fd987193");

FF ($C, $D, $A, $B, $words[14 + ($i \* 16)], 17, "a679438e");

FF ($B, $C, $D, $A, $words[15 + ($i \* 16)], 22, "49b40821");

GG ($A, $B, $C, $D, $words[1 + ($i \* 16)], 5, "f61e2562");

GG ($D, $A, $B, $C, $words[6 + ($i \* 16)], 9, "c040b340");

GG ($C, $D, $A, $B, $words[11 + ($i \* 16)], 14, "265e5a51");

GG ($B, $C, $D, $A, $words[0 + ($i \* 16)], 20, "e9b6c7aa");

GG ($A, $B, $C, $D, $words[5 + ($i \* 16)], 5, "d62f105d");

GG ($D, $A, $B, $C, $words[10 + ($i \* 16)], 9, "02441453");

GG ($C, $D, $A, $B, $words[15 + ($i \* 16)], 14, "d8a1e681");

GG ($B, $C, $D, $A, $words[4 + ($i \* 16)], 20, "e7d3fbc8");

GG ($A, $B, $C, $D, $words[9 + ($i \* 16)], 5, "21e1cde6");

GG ($D, $A, $B, $C, $words[14 + ($i \* 16)], 9, "c33707d6");

GG ($C, $D, $A, $B, $words[3 + ($i \* 16)], 14, "f4d50d87");

GG ($B, $C, $D, $A, $words[8 + ($i \* 16)], 20, "455a14ed");

GG ($A, $B, $C, $D, $words[13 + ($i \* 16)], 5, "a9e3e905");

GG ($D, $A, $B, $C, $words[2 + ($i \* 16)], 9, "fcefa3f8");

GG ($C, $D, $A, $B, $words[7 + ($i \* 16)], 14, "676f02d9");

GG ($B, $C, $D, $A, $words[12 + ($i \* 16)], 20, "8d2a4c8a");

HH ($A, $B, $C, $D, $words[5 + ($i \* 16)], 4, "fffa3942");

HH ($D, $A, $B, $C, $words[8 + ($i \* 16)], 11, "8771f681");

HH ($C, $D, $A, $B, $words[11 + ($i \* 16)], 16, "6d9d6122");

HH ($B, $C, $D, $A, $words[14 + ($i \* 16)], 23, "fde5380c");

HH ($A, $B, $C, $D, $words[1 + ($i \* 16)], 4, "a4beea44");

HH ($D, $A, $B, $C, $words[4 + ($i \* 16)], 11, "4bdecfa9");

HH ($C, $D, $A, $B, $words[7 + ($i \* 16)], 16, "f6bb4b60");

HH ($B, $C, $D, $A, $words[10 + ($i \* 16)], 23, "bebfbc70");

HH ($A, $B, $C, $D, $words[13 + ($i \* 16)], 4, "289b7ec6");

HH ($D, $A, $B, $C, $words[0 + ($i \* 16)], 11, "eaa127fa");

HH ($C, $D, $A, $B, $words[3 + ($i \* 16)], 16, "d4ef3085");

HH ($B, $C, $D, $A, $words[6 + ($i \* 16)], 23, "04881d05");

HH ($A, $B, $C, $D, $words[9 + ($i \* 16)], 4, "d9d4d039");

HH ($D, $A, $B, $C, $words[12 + ($i \* 16)], 11, "e6db99e5");

HH ($C, $D, $A, $B, $words[15 + ($i \* 16)], 16, "1fa27cf8");

HH ($B, $C, $D, $A, $words[2 + ($i \* 16)], 23, "c4ac5665");

II ($A, $B, $C, $D, $words[0 + ($i \* 16)], 6, "f4292244");

II ($D, $A, $B, $C, $words[7 + ($i \* 16)], 10, "432aff97");

II ($C, $D, $A, $B, $words[14 + ($i \* 16)], 15, "ab9423a7");

II ($B, $C, $D, $A, $words[5 + ($i \* 16)], 21, "fc93a039");

II ($A, $B, $C, $D, $words[12 + ($i \* 16)], 6, "655b59c3");

II ($D, $A, $B, $C, $words[3 + ($i \* 16)], 10, "8f0ccc92");

II ($C, $D, $A, $B, $words[10 + ($i \* 16)], 15, "ffeff47d");

II ($B, $C, $D, $A, $words[1 + ($i \* 16)], 21, "85845dd1");

II ($A, $B, $C, $D, $words[8 + ($i \* 16)], 6, "6fa87e4f");

II ($D, $A, $B, $C, $words[15 + ($i \* 16)], 10, "fe2ce6e0");

II ($C, $D, $A, $B, $words[6 + ($i \* 16)], 15, "a3014314");

II ($B, $C, $D, $A, $words[13 + ($i \* 16)], 21, "4e0811a1");

II ($A, $B, $C, $D, $words[4 + ($i \* 16)], 6, "f7537e82");

II ($D, $A, $B, $C, $words[11 + ($i \* 16)], 10, "bd3af235");

II ($C, $D, $A, $B, $words[2 + ($i \* 16)], 15, "2ad7d2bb");

II ($B, $C, $D, $A, $words[9 + ($i \* 16)], 21, "eb86d391");

addVars($a, $b, $c, $d, $A, $B, $C, $D);

}

$MD5 = '';

foreach (array($a, $b, $c, $d) as $x) {

$MD5 .= implode('', array\_reverse(str\_split(leftpad($x, 8), 2)));

}

return $MD5;

}

function hexbin($str){

$hexbinmap = array("0" => "0000"

, "1" => "0001"

, "2" => "0010"

, "3" => "0011"

, "4" => "0100"

, "5" => "0101"

, "6" => "0110"

, "7" => "0111"

, "8" => "1000"

, "9" => "1001"

, "A" => "1010"

, "a" => "1010"

, "B" => "1011"

, "b" => "1011"

, "C" => "1100"

, "c" => "1100"

, "D" => "1101"

, "d" => "1101"

, "E" => "1110"

, "e" => "1110"

, "F" => "1111"

, "f" => "1111");

$bin = "";

for ($i = 0; $i < strlen($str); $i++)

{

$bin .= $hexbinmap[$str[$i]];

}

$bin = ltrim($bin, '0');

// echo "Original: ".$str." New: ".$bin."<br />";

return $bin;

}

function strhex($str){

$hex = "";

for ($i = 0; $i < strlen($str); $i++)

{

$hex = $hex.leftpad(dechex(ord($str[$i])), 2);

}

return $hex;

}

function init($string){

$len = strlen($string) \* 8;

$hex = strhex($string);

$bin = leftpad(hexbin($hex), $len);

$padded = pad($bin);

$padded = pad($padded, 1, $len);

$block = str\_split($padded, 32);

foreach ($block as &$b) {

$b = implode('', array\_reverse(str\_split($b, 8)));

}

return $block;

}

function pad($bin, $type=0, $len = 0){

if($type == 0){

$bin = $bin."1";

$buff = strlen($bin) % 512;

if($buff != 448){

while(strlen($bin) % 512 != 448){

$bin = $bin."0";

}

}

}

elseif($type == 1){

$bLen = leftpad(decbin($len), 64);

$bin .= implode('', array\_reverse(str\_split($bLen, 8)));

}

return $bin;

}

function F($X, $Y, $Z){

$X = hexdec($X);

$Y = hexdec($Y);

$Z = hexdec($Z);

$calc = (($X & $Y) | ((~ $X) & $Z));

return $calc;

}

function G($X, $Y, $Z){

$X = hexdec($X);

$Y = hexdec($Y);

$Z = hexdec($Z);

$calc = (($X & $Z) | ($Y & (~ $Z)));

return $calc;

}

function H($X, $Y, $Z){

$X = hexdec($X);

$Y = hexdec($Y);

$Z = hexdec($Z);

$calc = ($X ^ $Y ^ $Z);

return $calc;

}

function I($X, $Y, $Z){

$X = hexdec($X);

$Y = hexdec($Y);

$Z = hexdec($Z);

$calc = ($Y ^ ($X | (~ $Z))) ;

return $calc;

}

$A - hex, $B - hex, $C - hex, $D - hex (F - dec)

$M - binary

$s - decimal

$t - hex

\*/

function FF(&$A, $B, $C, $D, $M, $s, $t){

$A = hexdec($A);

$t = hexdec($t);

$M = bindec($M);

$A = ($A + F($B, $C, $D) + $M + $t) & 0xffffffff;

$A = rotate($A, $s);

$A = dechex((hexdec($B) + hexdec($A)) & 0xffffffff);

}

function GG(&$A, $B, $C, $D, $M, $s, $t){

$A = hexdec($A);

$t = hexdec($t);

$M = bindec($M);

$A = ($A + G($B, $C, $D) + $M + $t) & 0xffffffff;

$A = rotate($A, $s);

$A = dechex((hexdec($B) + hexdec($A)) & 0xffffffff);

}

function HH(&$A, $B, $C, $D, $M, $s, $t){

$A = hexdec($A);

$t = hexdec($t);

$M = bindec($M);

$A = ($A + H($B, $C, $D) + $M + $t) & 0xffffffff;

$A = rotate($A, $s);

$A = dechex((hexdec($B) + hexdec($A)) & 0xffffffff);

}

function II(&$A, $B, $C, $D, $M, $s, $t){

$A = hexdec($A);

$t = hexdec($t);

$M = bindec($M);

$A = ($A + I($B, $C, $D) + $M + $t) & 0xffffffff;

$A = rotate($A, $s);

$A = dechex((hexdec($B) + hexdec($A)) & 0xffffffff);

}

function rotate ($decimal, $bits) {

return dechex((($decimal << $bits) | ($decimal >> (32 - $bits))) & 0xffffffff);

}

function addVars(&$a, &$b, &$c, &$d, $A, $B, $C, $D){

$A = hexdec($A);

$B = hexdec($B);

$C = hexdec($C);

$D = hexdec($D);

$aa = hexdec($a);

$bb = hexdec($b);

$cc = hexdec($c);

$dd = hexdec($d);

$aa = ($aa + $A) & 0xffffffff;

$bb = ($bb + $B) & 0xffffffff;

$cc = ($cc + $C) & 0xffffffff;

$dd = ($dd + $D) & 0xffffffff;

$a = dechex($aa);

$b = dechex($bb);

$c = dechex($cc);

$d = dechex($dd);

}

function leftpad($needs\_padding, $alignment)

{

if (strlen($needs\_padding) % $alignment) {

$pad\_amount = $alignment - strlen($needs\_padding) % $alignment;

$left\_pad = implode('', array\_fill(0, $pad\_amount, '0'));

$needs\_padding = $left\_pad . $needs\_padding;

}

return $needs\_padding;

}

**Висновок:** в ході лабораторної роботи було освоєно алгоритм шифрування MD5. Отриманні знання про важливість захисту даних. Покращено навички роботи з мовами програмування PHP, JS,JQuery.