**Звіт**

**До практичної роботи**

**з дисципліни «Основи інформаційної безпеки»**

Студента Київського Національного Університету ім. Тараса Шевченко

Переверзева Олексія Сергійовича

Факультет інформаційних технологій

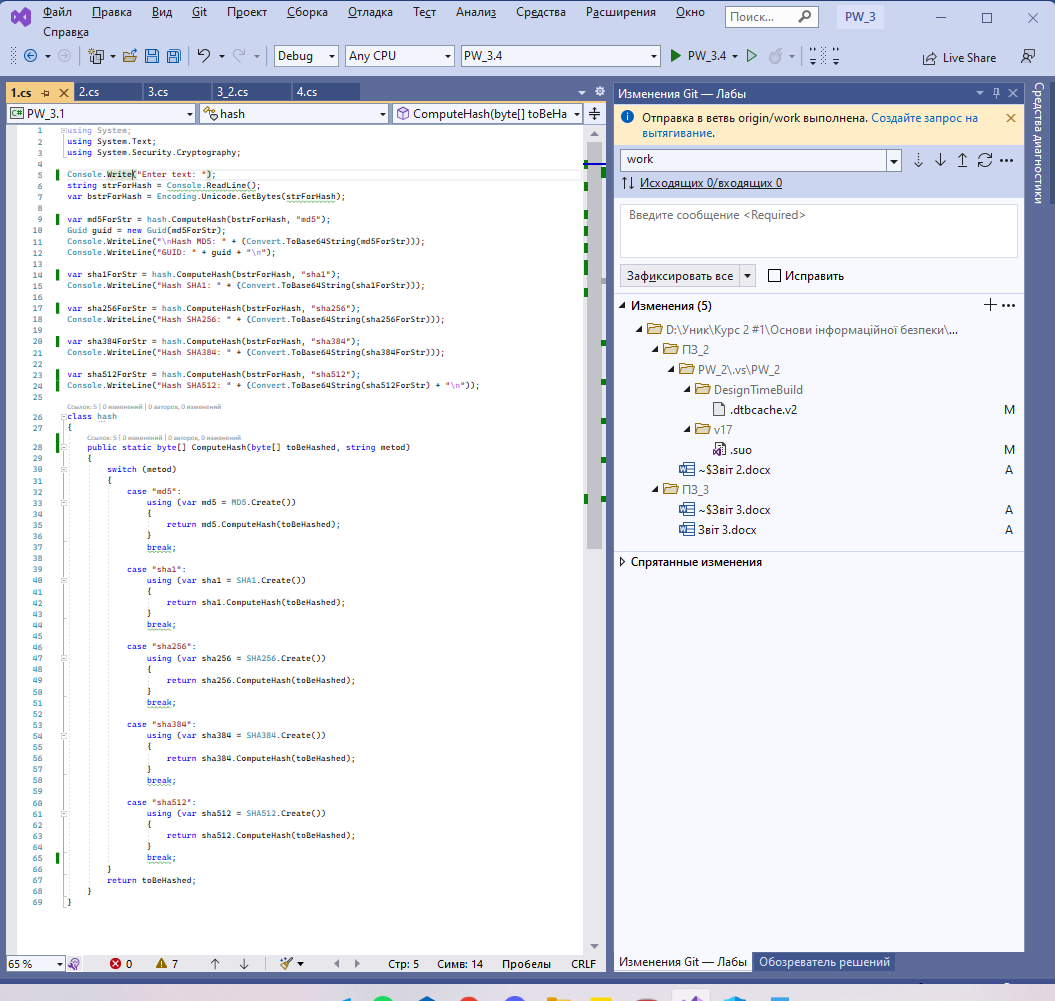
Група МІТ-21



**Практичне заняття №3**

**Тема:** Хеш-функції та перевірка цілісності інформації

1. Написати програму, яка обчислює хеш-коди за всіма відомими алгоритмами для заданих даних. Порівняти розміри хеш-кодів та значення для однакових та різних даних.



Спочатку вводимо будь-яке повідомлення з клавіатури для подальшого обчислення хеш-кодів:

Console.Write("Enter text: ");

string strForHash = Console.ReadLine();

var bstrForHash = Encoding.Unicode.GetBytes(strForHash);

Відповідно далі створимо відповідні функції для обчислення хеш-кодів. В моєму випадку я використав оператор вибору switch , тому я помістив все в одну функцію, але передаю змінну, яка містить назву алгоритму, який буде використовуватись для обчислення хеш-кодів:

public static byte[] ComputeHash(byte[] toBeHashed, string metod)

{

switch (metod)

{

case "md5":

using (var md5 = MD5.Create())

{

return md5.ComputeHash(toBeHashed);

}

break;

case "sha1":

using (var sha1 = SHA1.Create())

{

return sha1.ComputeHash(toBeHashed);

}

break;

case "sha256":

using (var sha256 = SHA256.Create())

{

return sha256.ComputeHash(toBeHashed);

}

break;

case "sha384":

using (var sha384 = SHA384.Create())

{

return sha384.ComputeHash(toBeHashed);

}

break;

case "sha512":

using (var sha512 = SHA512.Create())

{

return sha512.ComputeHash(toBeHashed);

}

break;

}

return toBeHashed;//для того щоб функція працювала //повертаю будь-яке значення, але цей сценарій ніколи не //виконується

}

Відповідно далі програма виводить обчислені хеш-коди в консоль за кожним алгоритмом, викликаючи функцію та передаючи в неї відповідні значення:

MD5:

var md5ForStr = hash.ComputeHash(bstrForHash, "md5");

Guid guid = new Guid(md5ForStr);//також можемо зберегти результат як //екземпляр структури GUID

Console.WriteLine("\nHash MD5: " + (Convert.ToBase64String(md5ForStr)));

Console.WriteLine("GUID: " + guid + "\n");

SHA1:

var sha1ForStr = hash.ComputeHash(bstrForHash, "sha1");

Console.WriteLine("Hash SHA1: " + (Convert.ToBase64String(sha1ForStr)));

SHA256:

var sha256ForStr = hash.ComputeHash(bstrForHash, "sha256");

Console.WriteLine("Hash SHA256: " + (Convert.ToBase64String(sha256ForStr)));

SHA384:

var sha384ForStr = hash.ComputeHash(bstrForHash, "sha384");

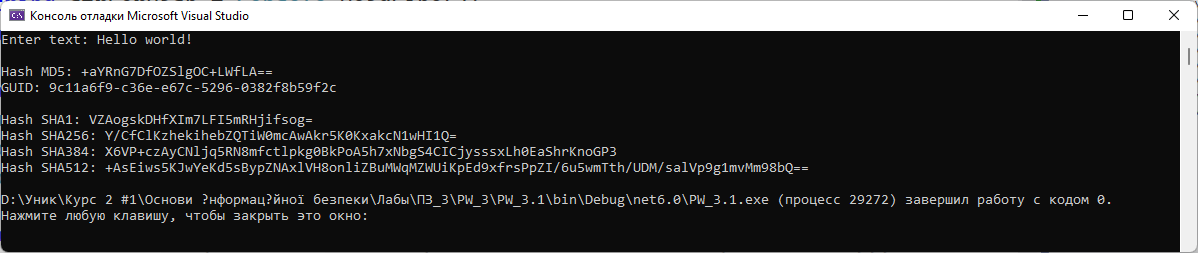
Console.WriteLine("Hash SHA384: " + (Convert.ToBase64String(sha384ForStr)));

SHA512:

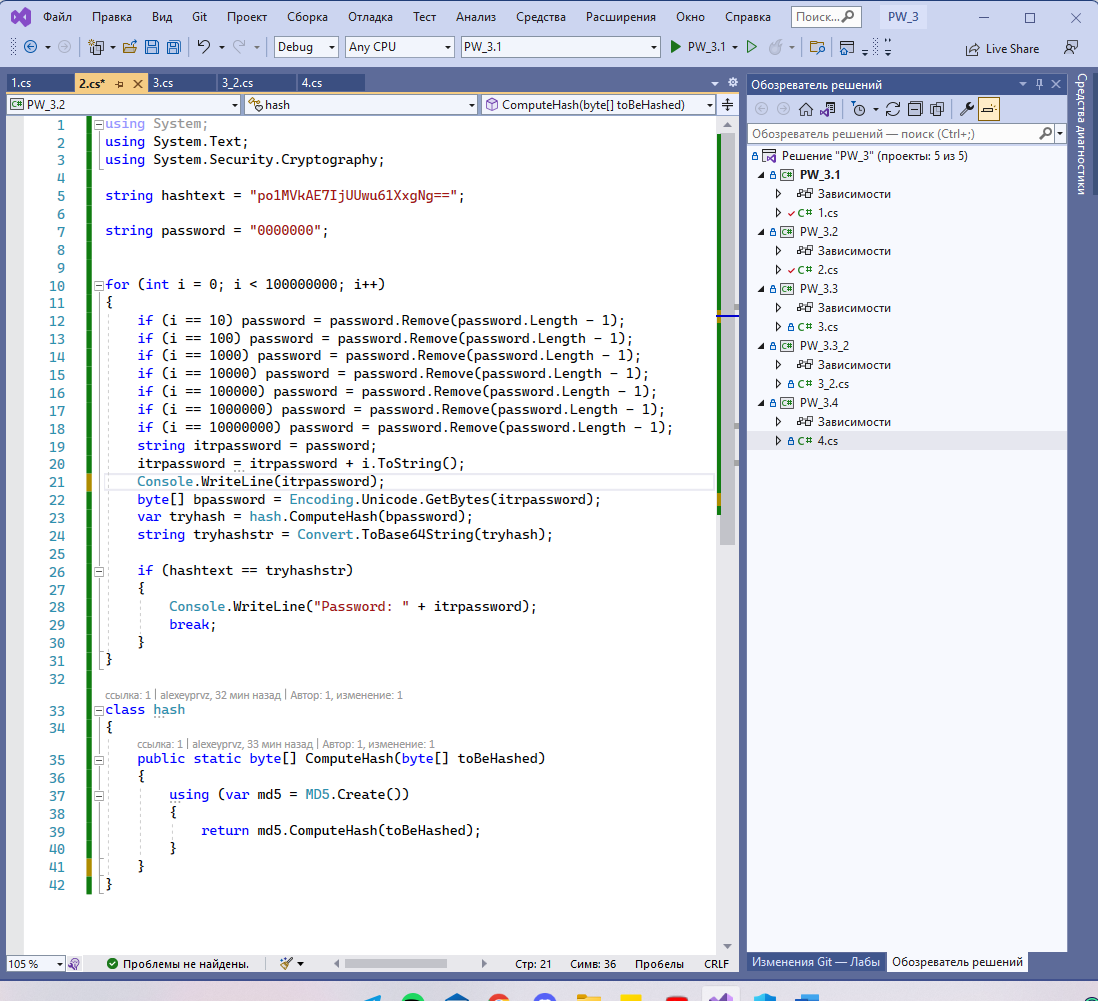
var sha512ForStr = hash.ComputeHash(bstrForHash, "sha512");

Console.WriteLine("Hash SHA512: " + (Convert.ToBase64String(sha512ForStr)));

Виконання програми:



1. Відомо, що користувач використав для пароля довжиною 8 символів лише цифри. Шляхом пасивного прослуховування мережі отримано MD5 хеш-код цього пароля: {564c8da6-0440-88ec-d453-0bbad57c6036} та po1MVkAE7IjUUwu61XxgNg==. Відновити пароль користувача та зробити висновки про надійність такого пароля.



Для початку внесемо потрібні нам дані, а саме отриманий MD5 хеш-код у змінну:

string hashtext = "po1MVkAE7IjUUwu61XxgNg==";

Механізм підбору пароля:

* Знаючи, що пароль складається з цифер, ми можемо використати цикл для проходження по порядку по всім значенням:

for (int i = 0; i < 100000000; i++)

* Але проблема в тому, що перед значеннями, які мають менше восьми цифер, повинні бути нулі, тому введемо змінну з нулями і будемо туди дописувати наші значення, тобто виконувати конкатенацію рядків:

string password = "0000000";

string itrpassword = password;

itrpassword = itrpassword + i.ToString();

* Відповідно при додаванні нового розряду ми будемо прибирати по одному нулю:

if (i == 10) password = password.Remove(password.Length - 1);

if (i == 100) password = password.Remove(password.Length - 1);

if (i == 1000) password = password.Remove(password.Length - 1);

if (i == 10000) password = password.Remove(password.Length - 1);

if (i == 100000) password = password.Remove(password.Length - 1);

if (i == 1000000) password = password.Remove(password.Length - 1);

if (i == 10000000) password = password.Remove(password.Length - 1);

* Далі перетворюємо пароль поточної ітерації у послідовність байтів, після цього обчислюємо хеш-код і перетворюємо його на рядок:

byte[] bpassword = Encoding.Unicode.GetBytes(itrpassword);

var tryhash = hash.ComputeHash(bpassword);

string tryhashstr = Convert.ToBase64String(tryhash);

* В кінці перевіряємо чи співпадає наш хеш-код з хеш-кодом поточної ітерації:

if (hashtext == tryhashstr)

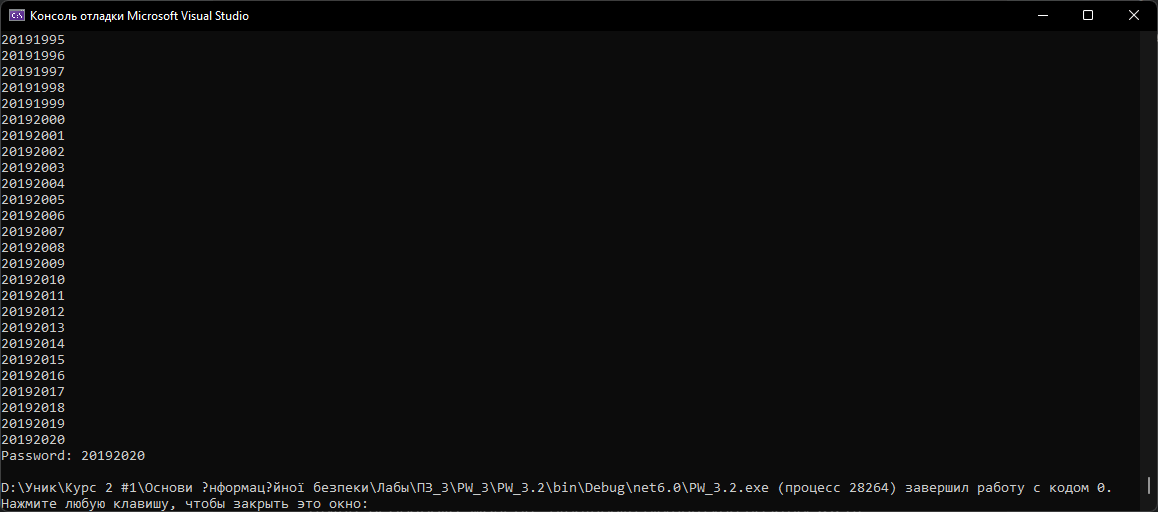
{

Console.WriteLine("Password: " + itrpassword);

break;

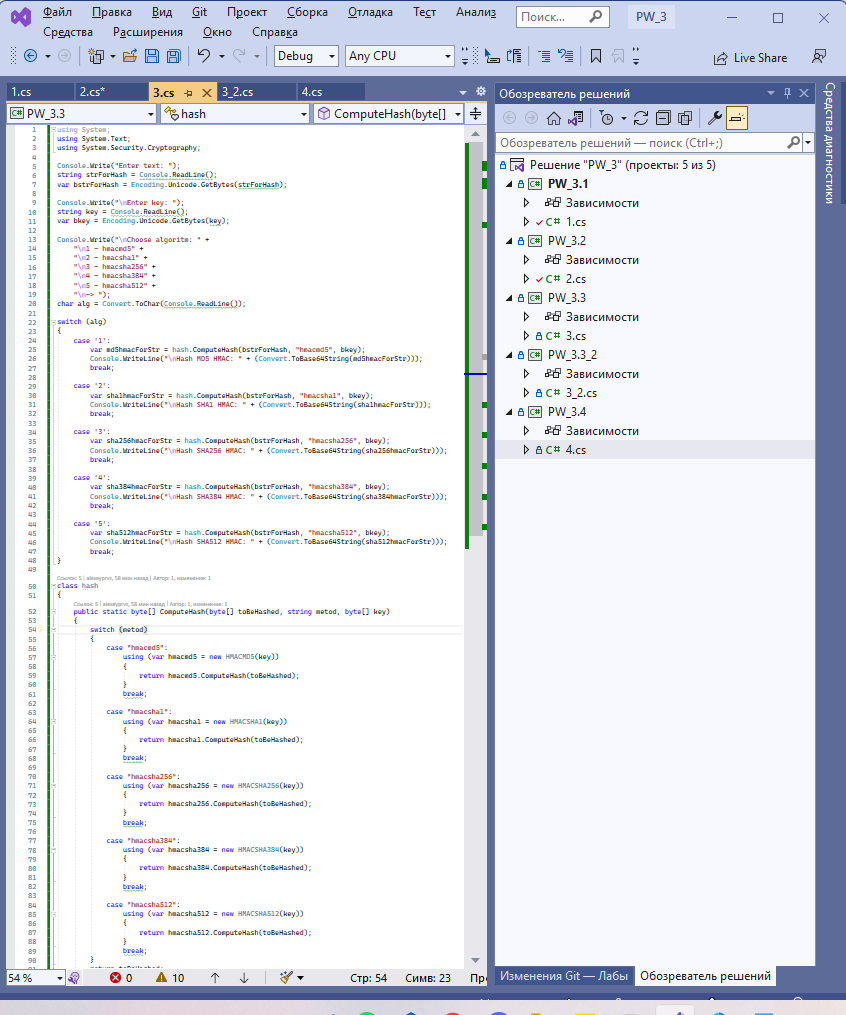
}

* Цикл і відповідно програма зупиняться, перед цим надрукувавши підібраний пароль:



Зачекавши приблизно 15-20 хвилин, ми отримали пароль **20192020.** Отже можна стверджувати, що такий пароль не є досить надійним, враховуючи його складність і час підбору.

1. Написати програму для обчислення хеш-коду автентифікації повідомлення а також реалізувати можливість перевірки автентичності повідомлення.



Так само спочатку вводимо повідомлення для його хешування та перетворюємо на послідовнімть байтів:

Console.Write("Enter text: ");

string strForHash = Console.ReadLine();

var bstrForHash = Encoding.Unicode.GetBytes(strForHash);

Далі вводимо ключ, який використовується в автентифікованому хешуванні:

Console.Write("\nEnter key: ");

string key = Console.ReadLine();

var bkey = Encoding.Unicode.GetBytes(key);

Аналогічно до 1 завдання виглядає структура функції, тільки звісно ще передаємо ключ у функцію:

class hash

{

public static byte[] ComputeHash(byte[] toBeHashed, string metod, byte[] key)

{

switch (metod)

{

case "hmacmd5":

using (var hmacmd5 = new HMACMD5(key))

{

return hmacmd5.ComputeHash(toBeHashed);

}

break;

case "hmacsha1":

using (var hmacsha1 = new HMACSHA1(key))

{

return hmacsha1.ComputeHash(toBeHashed);

}

break;

case "hmacsha256":

using (var hmacsha256 = new HMACSHA256(key))

{

return hmacsha256.ComputeHash(toBeHashed);

}

break;

case "hmacsha384":

using (var hmacsha384 = new HMACSHA384(key))

{

return hmacsha384.ComputeHash(toBeHashed);

}

break;

case "hmacsha512":

using (var hmacsha512 = new HMACSHA512(key))

{

return hmacsha512.ComputeHash(toBeHashed);

}

break;

}

return toBeHashed;

}

}

Аналогічно до першого завдання виводимо в консоль хеш-код, але додав можливість користувачу обрати алгоритм:

Console.Write("\nChoose algoritm: " +

"\n1 - hmacmd5" +

"\n2 - hmacsha1" +

"\n3 - hmacsha256" +

"\n4 - hmacsha384" +

"\n5 - hmacsha512" +

"\n-> ");

char alg = Convert.ToChar(Console.ReadLine());

switch (alg)

{

case '1':

var md5hmacForStr = hash.ComputeHash(bstrForHash, "hmacmd5", bkey);

Console.WriteLine("\nHash MD5 HMAC: " + (Convert.ToBase64String(md5hmacForStr)));

break;

case '2':

var sha1hmacForStr = hash.ComputeHash(bstrForHash, "hmacsha1", bkey);

Console.WriteLine("\nHash SHA1 HMAC: " + (Convert.ToBase64String(sha1hmacForStr)));

break;

case '3':

var sha256hmacForStr = hash.ComputeHash(bstrForHash, "hmacsha256", bkey);

Console.WriteLine("\nHash SHA256 HMAC: " + (Convert.ToBase64String(sha256hmacForStr)));

break;

case '4':

var sha384hmacForStr = hash.ComputeHash(bstrForHash, "hmacsha384", bkey);

Console.WriteLine("\nHash SHA384 HMAC: " + (Convert.ToBase64String(sha384hmacForStr)));

break;

case '5':

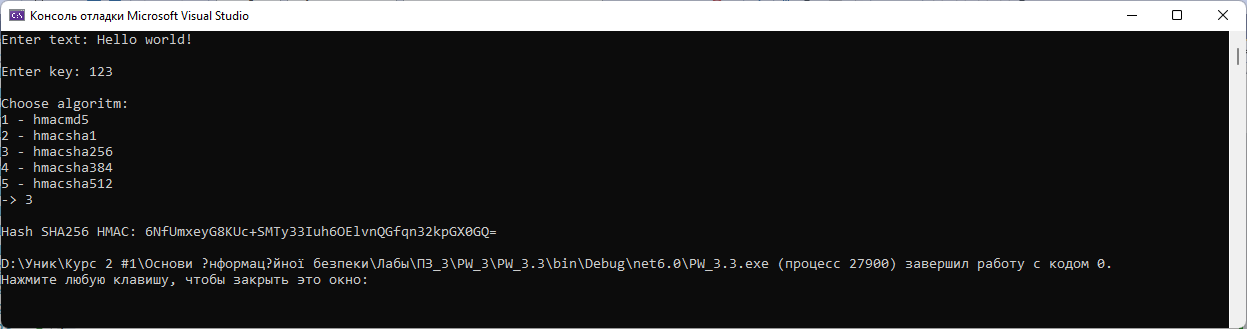
var sha512hmacForStr = hash.ComputeHash(bstrForHash, "hmacsha512", bkey);

Console.WriteLine("\nHash SHA512 HMAC: " + (Convert.ToBase64String(sha512hmacForStr)));

break;

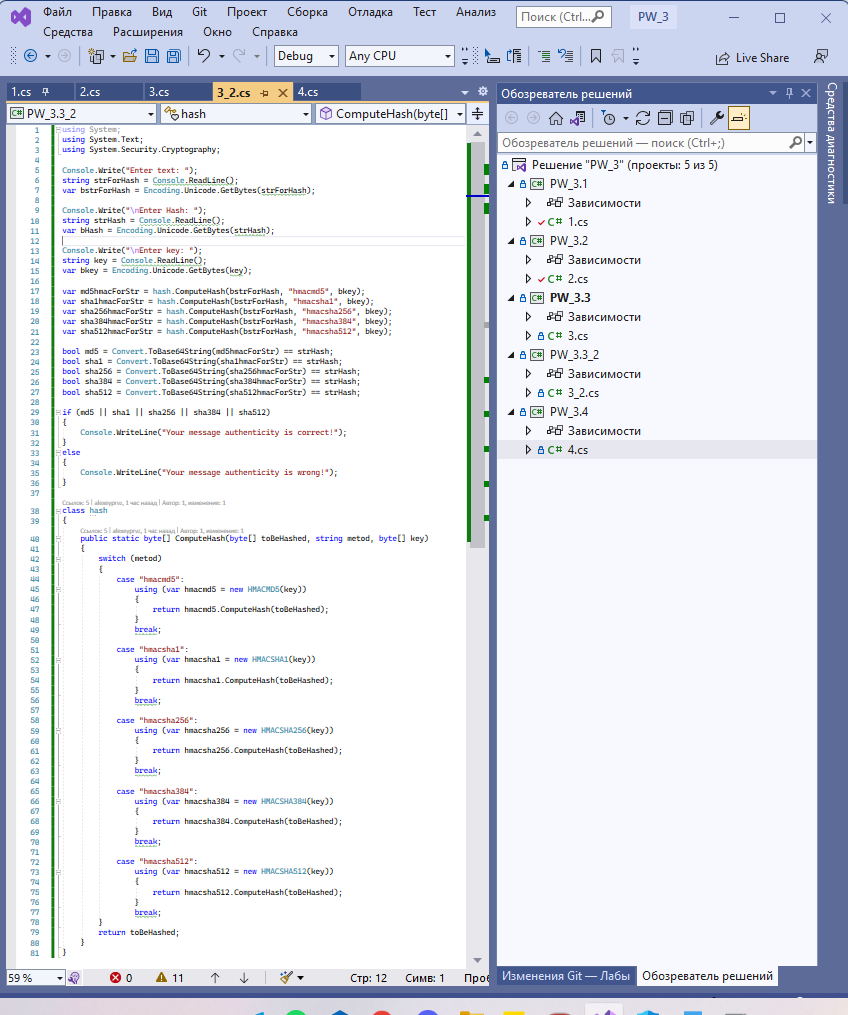
}

Виконання програми:



Аналогічно з іншими алгоритмами.

Також окремим проектом зробив перевірку автентичності повідомлення:



В цілому тут все теж саме, тому зупинюсь на відмінностях від попереднього коду:

* Відповідно, перевіряючи автентичність повідомлення, ми вводимо саме повідомлення і його хеш код, який ви наприклад отримали від іншого довіреного користувача по мережі. Маючи ключ, який зберігається у вас, ви також його вводите:

Console.Write("Enter text: ");

string strForHash = Console.ReadLine();

var bstrForHash = Encoding.Unicode.GetBytes(strForHash);

Console.Write("\nEnter Hash: ");

string strHash = Console.ReadLine();

var bHash = Encoding.Unicode.GetBytes(strHash);

Console.Write("\nEnter key: ");

string key = Console.ReadLine();

var bkey = Encoding.Unicode.GetBytes(key);

* Далі, щоб не турбувати користувача, обчислюємо хеш-код за всіма алгоритмами:

var md5hmacForStr = hash.ComputeHash(bstrForHash, "hmacmd5", bkey);

var sha1hmacForStr = hash.ComputeHash(bstrForHash, "hmacsha1", bkey);

var sha256hmacForStr = hash.ComputeHash(bstrForHash, "hmacsha256", bkey);

var sha384hmacForStr = hash.ComputeHash(bstrForHash, "hmacsha384", bkey);

var sha512hmacForStr = hash.ComputeHash(bstrForHash, "hmacsha512", bkey);

* Вводимо булеві змінні, в яких порівнюємо введений хеш-код в консоль та тільки що обчислений:

bool md5 = Convert.ToBase64String(md5hmacForStr) == strHash;

bool sha1 = Convert.ToBase64String(sha1hmacForStr) == strHash;

bool sha256 = Convert.ToBase64String(sha256hmacForStr) == strHash;

bool sha384 = Convert.ToBase64String(sha384hmacForStr) == strHash;

bool sha512 = Convert.ToBase64String(sha512hmacForStr) == strHash;

* В кінці перевіряємо, якщо один з варіантів вірний, то ми отримаємо повідомлення про коректну автентичність повідомлення, інакше негативне повідомлення:

if (md5 || sha1 || sha256 || sha384 || sha512)

{

Console.WriteLine("Your message authenticity is correct!");

}

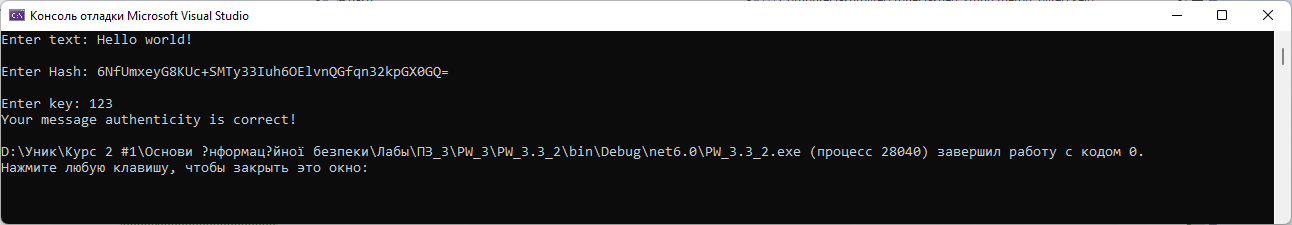
else

{

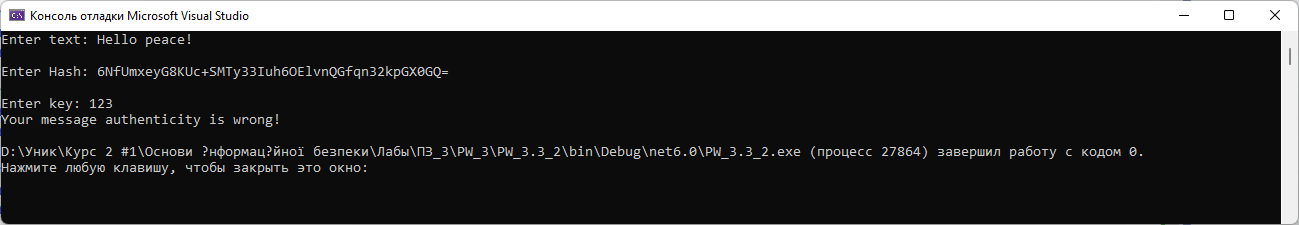
Console.WriteLine("Your message authenticity is wrong!");

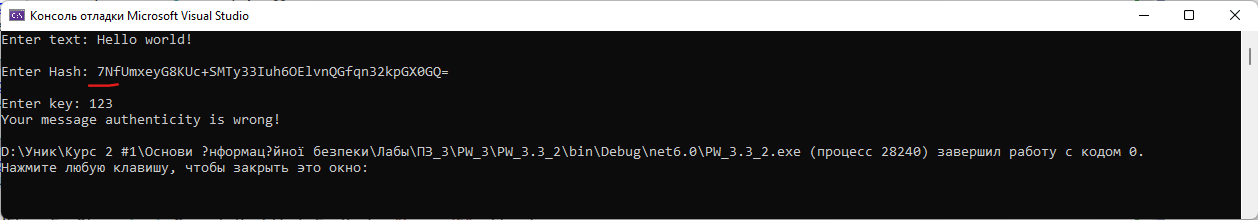
}

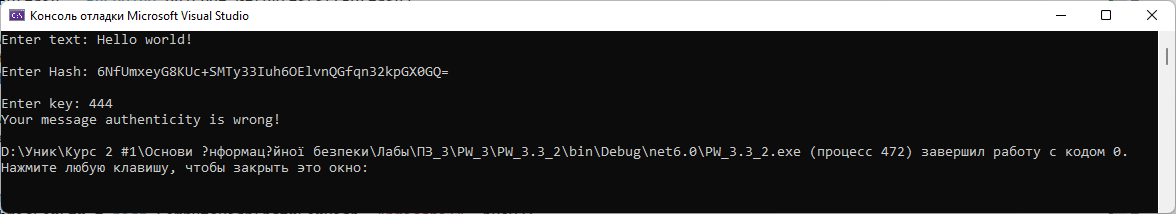
Виконання програми(використовуємо введені та отримані дані з минулої програми):



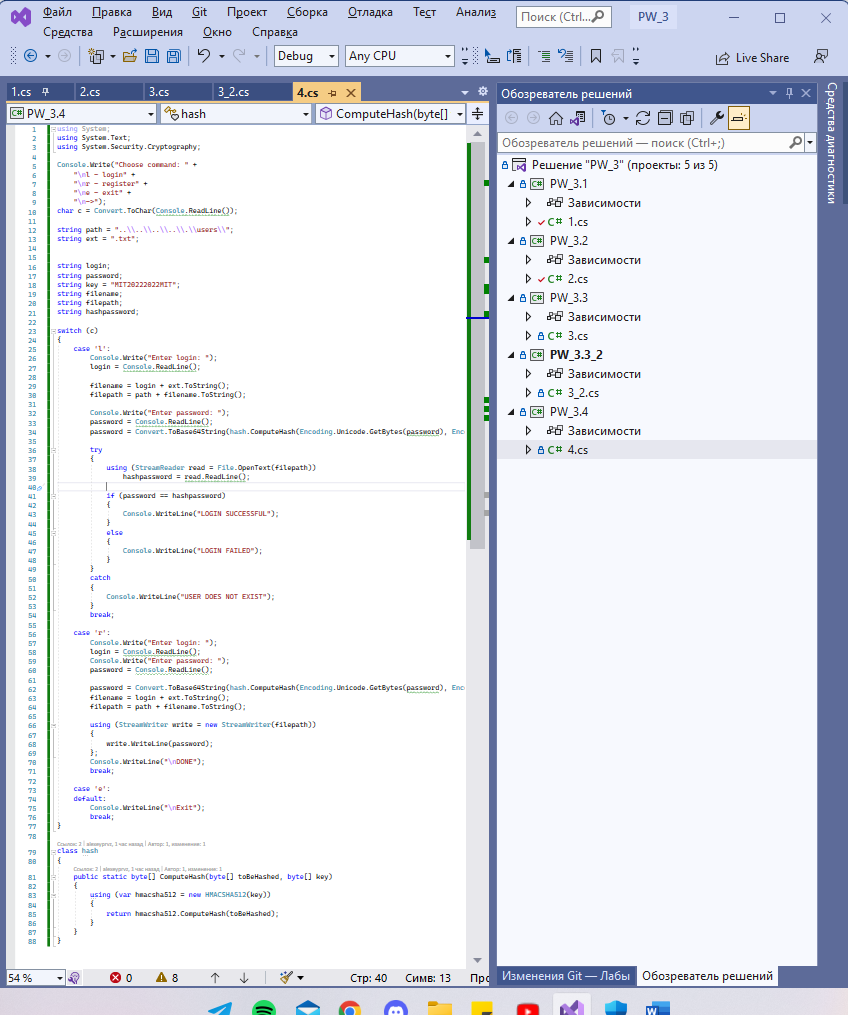
Ну і отримаємо інше повідомлення, якщо змінити будь-які дані(або в реальній ситуації вони були змінені третьою особою):







1. Написати програму для реєстрації користувача за логіном/паролем та авторизації шляхом співставлення відповідних логінів і паролів. Зберігання пароля у відкритому вигляді неприпустиме.



В цій програмі можна увійти в запис або зареєструватись:

Console.Write("Choose command: " +

"\nl - login" +

"\nr - register" +

"\ne - exit" +

"\n->");

Вказуємо шлях до папки, в якій будемо зберігати файли:

string path = "..\\..\\..\\..\\.\\users\\";

Обчислення хеш-коду відбувається за алгоритмом HMACSHA512:

class hash

{

public static byte[] ComputeHash(byte[] toBeHashed, byte[] key)

{

using (var hmacsha512 = new HMACSHA512(key))

{

return hmacsha512.ComputeHash(toBeHashed);

}

}

}

Також зберігаємо в коді ключ до функції хешування:

string key = "MIT20222022MIT";

Як відбувається реєстрація:

* Спочатку вводимо в консоль логін і пароль:

Console.Write("Enter login: ");

login = Console.ReadLine();

Console.Write("Enter password: ");

password = Console.ReadLine();

* Далі в ту ж змінну перезаписуємо отримане текстове представлення хеш-коду:

password = Convert.ToBase64String(hash.ComputeHash(Encoding.Unicode.GetBytes(password), Encoding.Unicode.GetBytes(key)));

* За допомогою конкатенації рядків створюємо ім’я файлу, а потім і шлях до нього:

string ext = ".txt";

filename = login + ext.ToString(); //”admin.txt”

filepath = path + filename.ToString();//"..\\..\\..\\..\\.\\users\\ admin.txt "

* Відкриваємо або створюємо файл відповідно до заданого логіну і записуємо в нього хеш-код паролю:

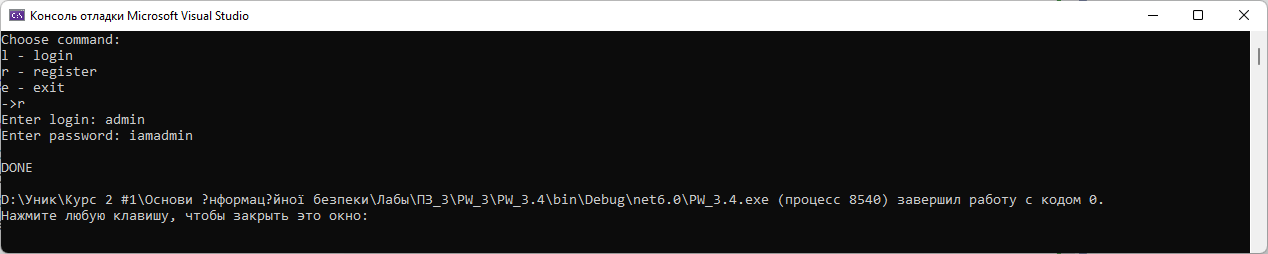
using (StreamWriter write = new StreamWriter(filepath))

{

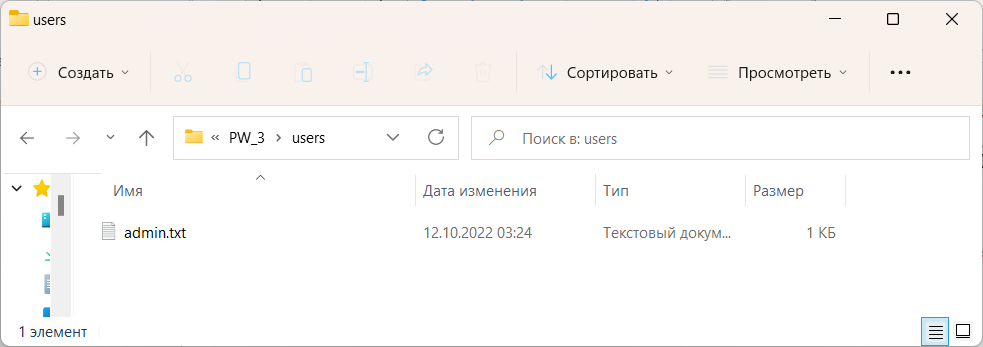
write.WriteLine(password);

};

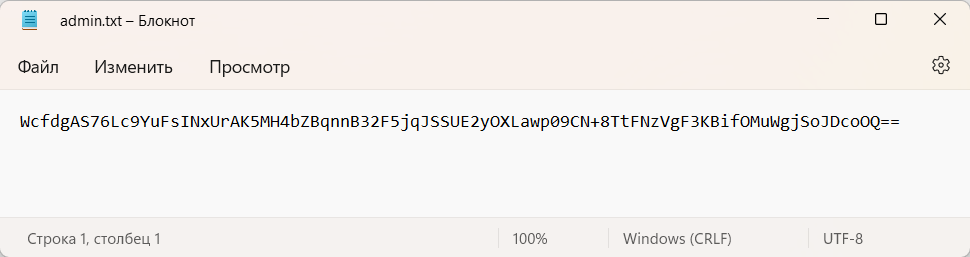
Виконання частини коду:



Створений файл:



Вміст файлу:



Як відбувається вхід:

* Думаю знову повторювати те саме немає сенсу, але відбувається за схожим принципом реєстрації, тільки далі ми створюємо хеш-код для введеного пароля в консоль і порівнюємо його з тим, що знаходиться у файлі, зчитавши його вміст:

Console.Write("Enter password: ");

password = Console.ReadLine();

password = Convert.ToBase64String(hash.ComputeHash(Encoding.Unicode.GetBytes(password), Encoding.Unicode.GetBytes(key)));

try

{

using (StreamReader read = File.OpenText(filepath))

hashpassword = read.ReadLine();

if (password == hashpassword)

{

Console.WriteLine("LOGIN SUCCESSFUL");

}

else

{

Console.WriteLine("LOGIN FAILED");

}

}

catch

{

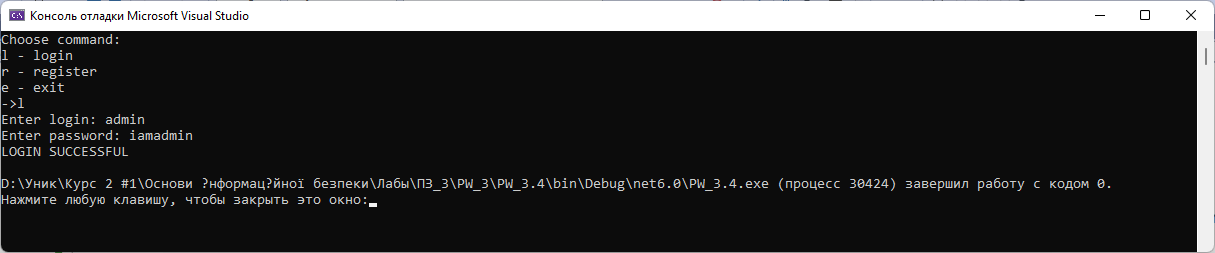
Console.WriteLine("USER DOES NOT EXIST");

}

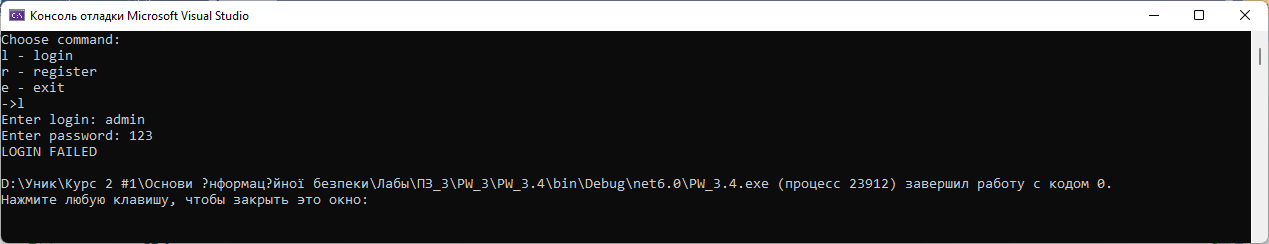
break;

* Щоб не виникало помилки при спробі прочитання файлу, якого не існує, тобто немає такого користувача, використовую try{ } catch{ } і тоді просто отримуємо повідомлення, що такого користувача немає. В іншому випадку вже просто відбувається перевірка правильності паролю.

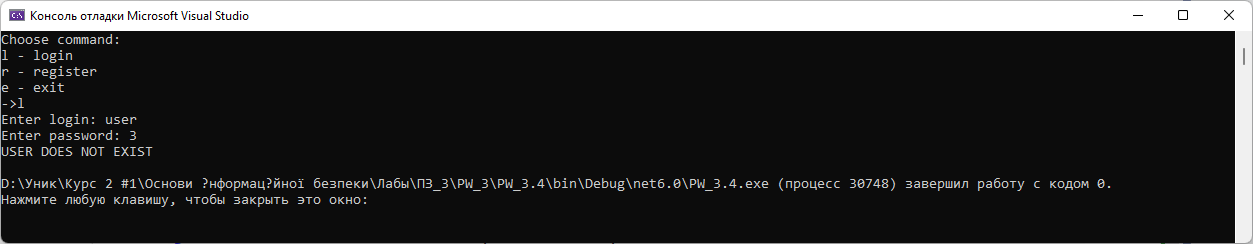
Виконання частини коду:



Ввівши невірний пароль:



Перевірка на існування користувача:



**Висновок:** в ході цієї практичної роботи ми навчились обчислювати хеш-коди за всіма алгоритмами та обчислення хеш-коду автентифікації повідомлення.