<https://github.com/bolshakovartem/osnovi-bezpeki.git>

Звіт

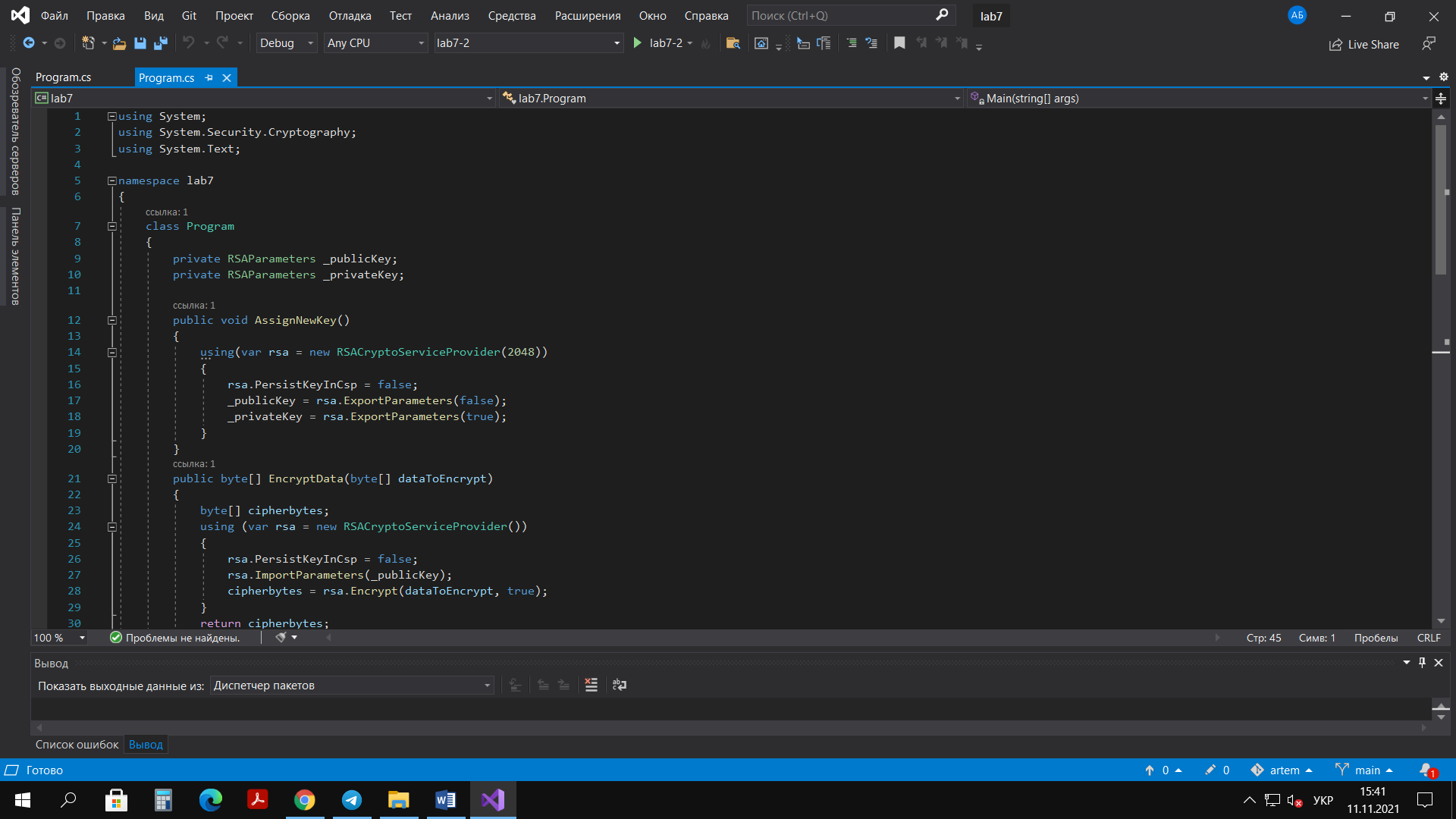
Лабораторна робота №7

Тема: «Асиметричне шифрування як засіб забезпечення

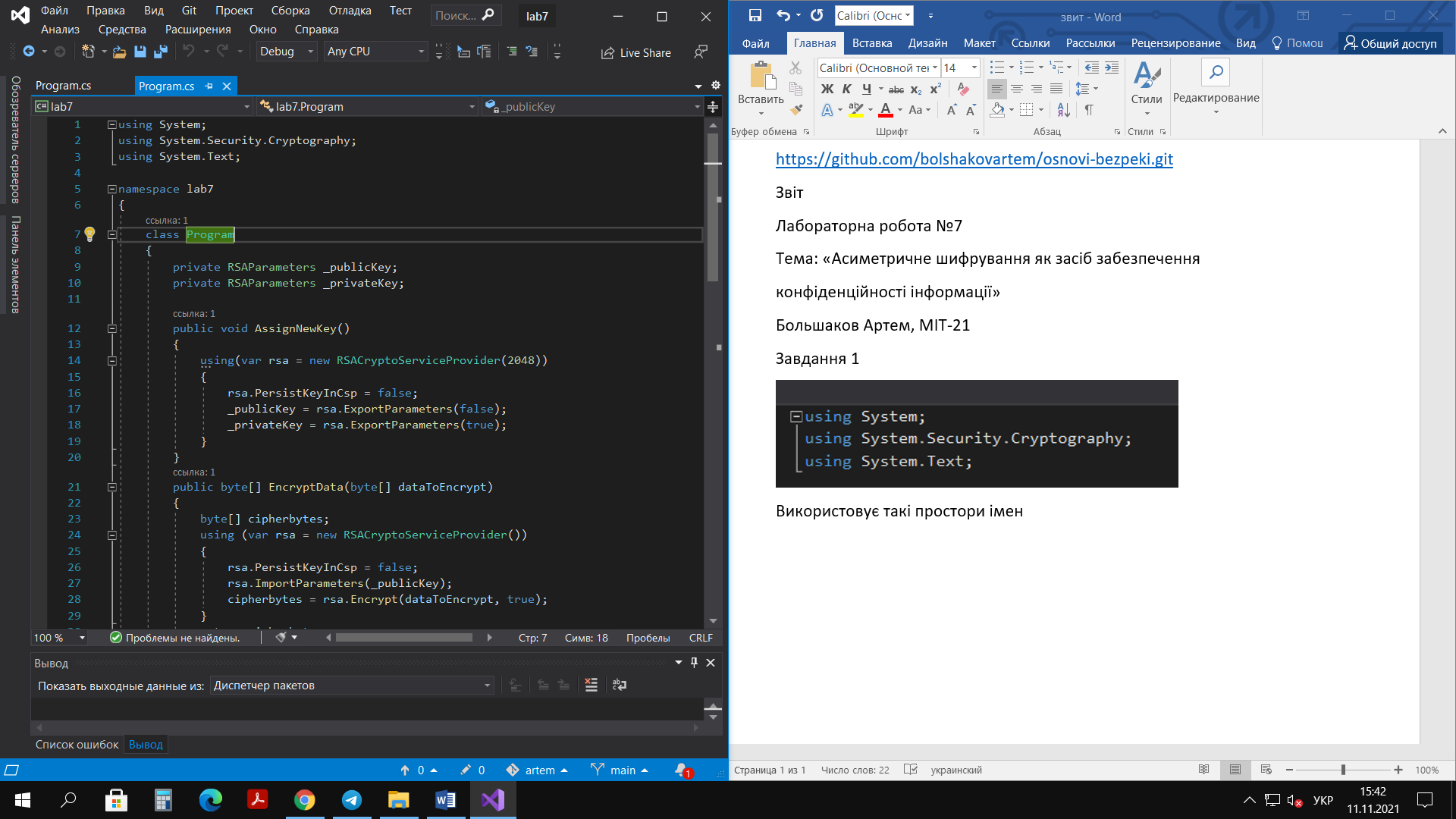
конфіденційності інформації»

Большаков Артем, МІТ-21

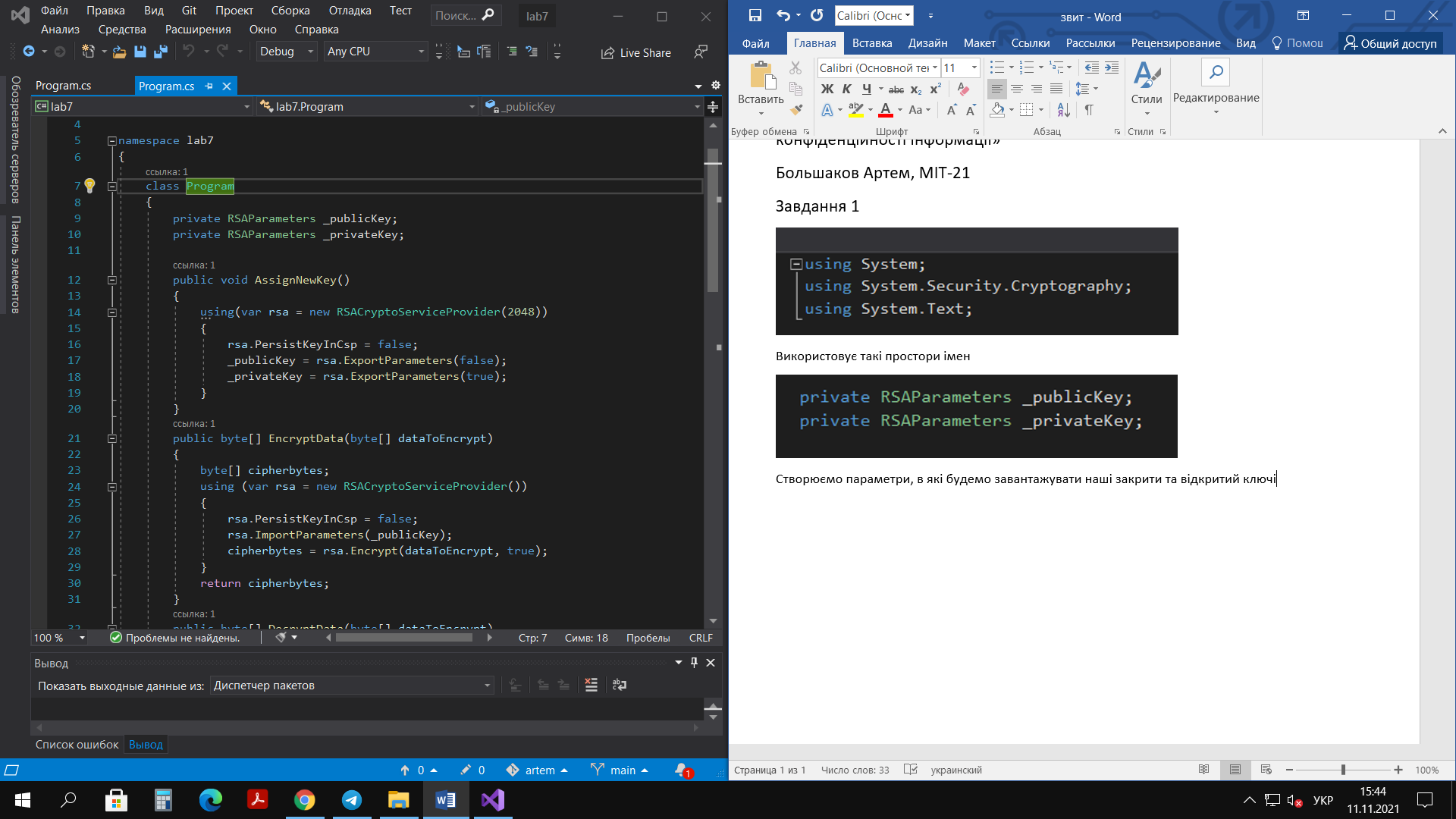
Завдання 1



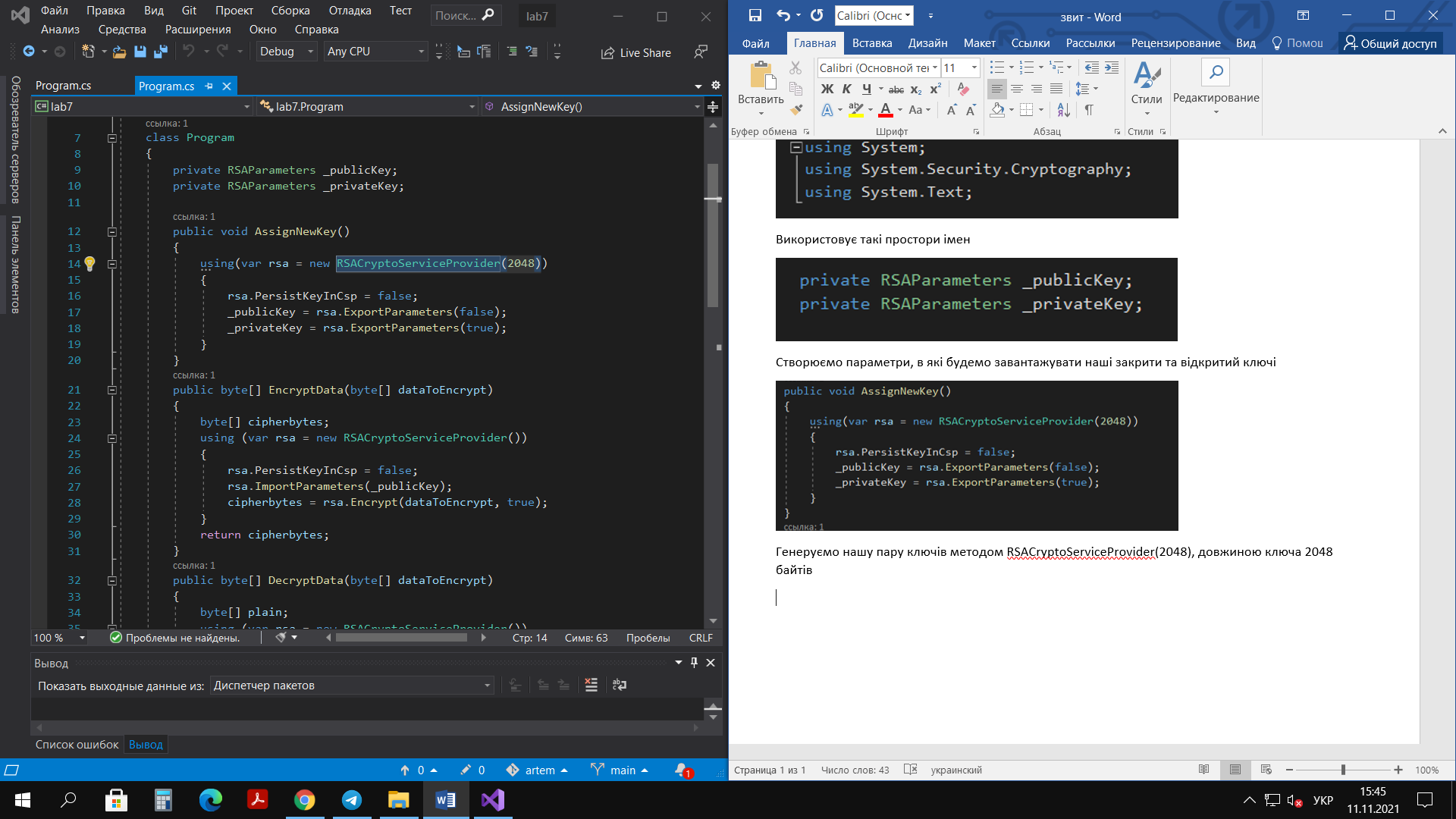
Використовує такі простори імен



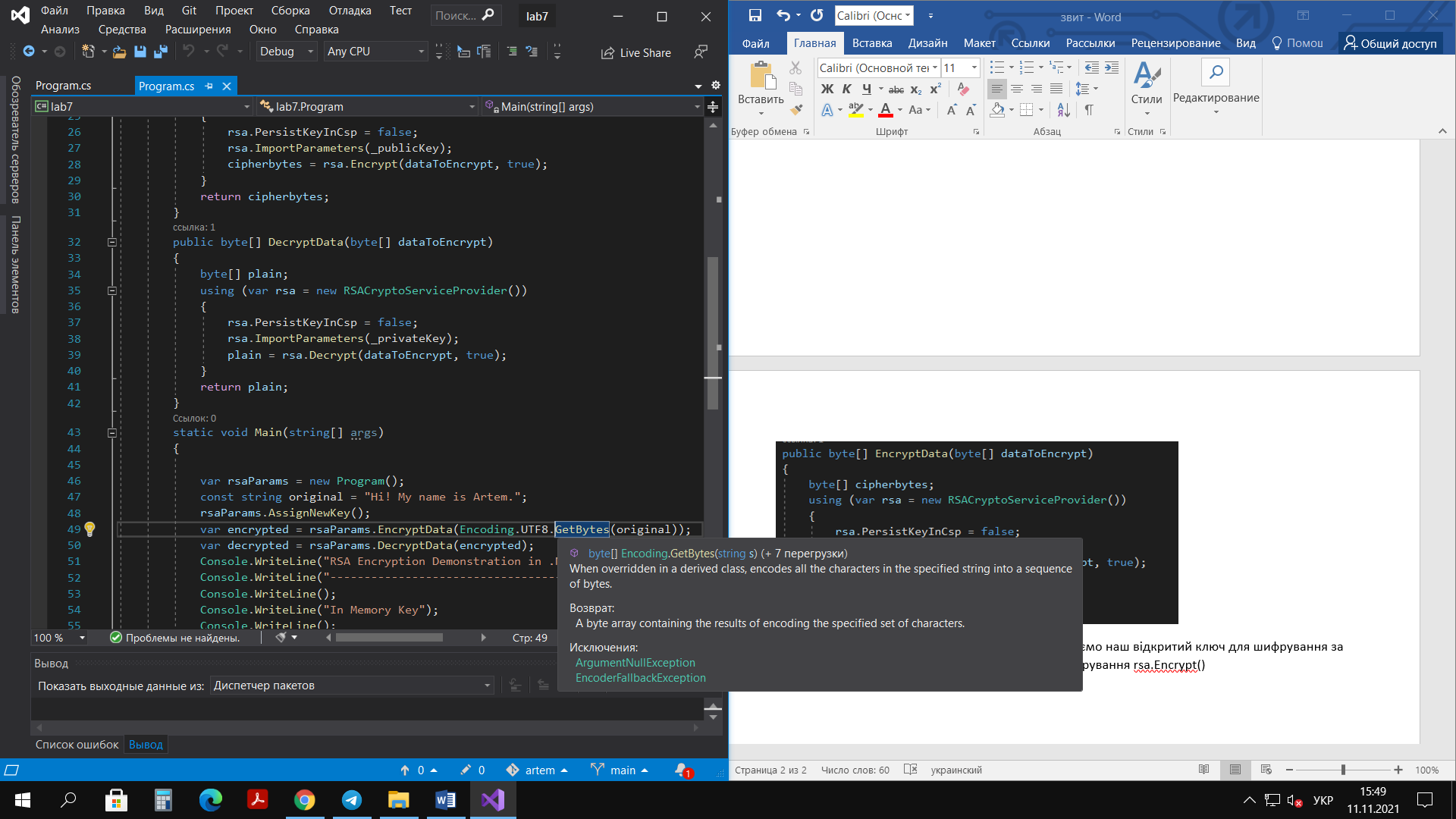
Створюємо параметри, в які будемо завантажувати наші закрити та відкритий ключі



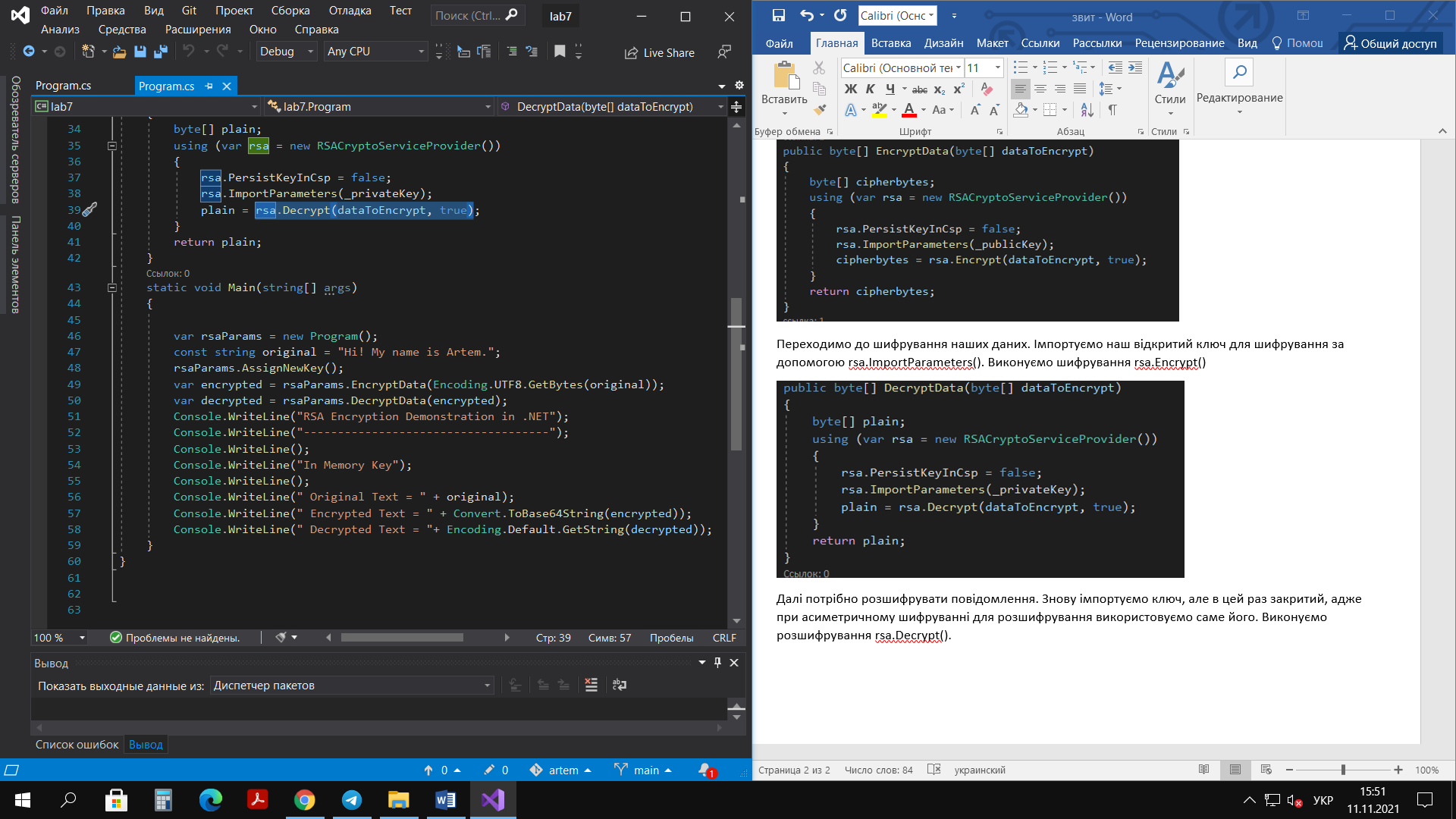
Генеруємо нашу пару ключів методом RSACryptoServiceProvider(2048), довжиною ключа 2048 байтів



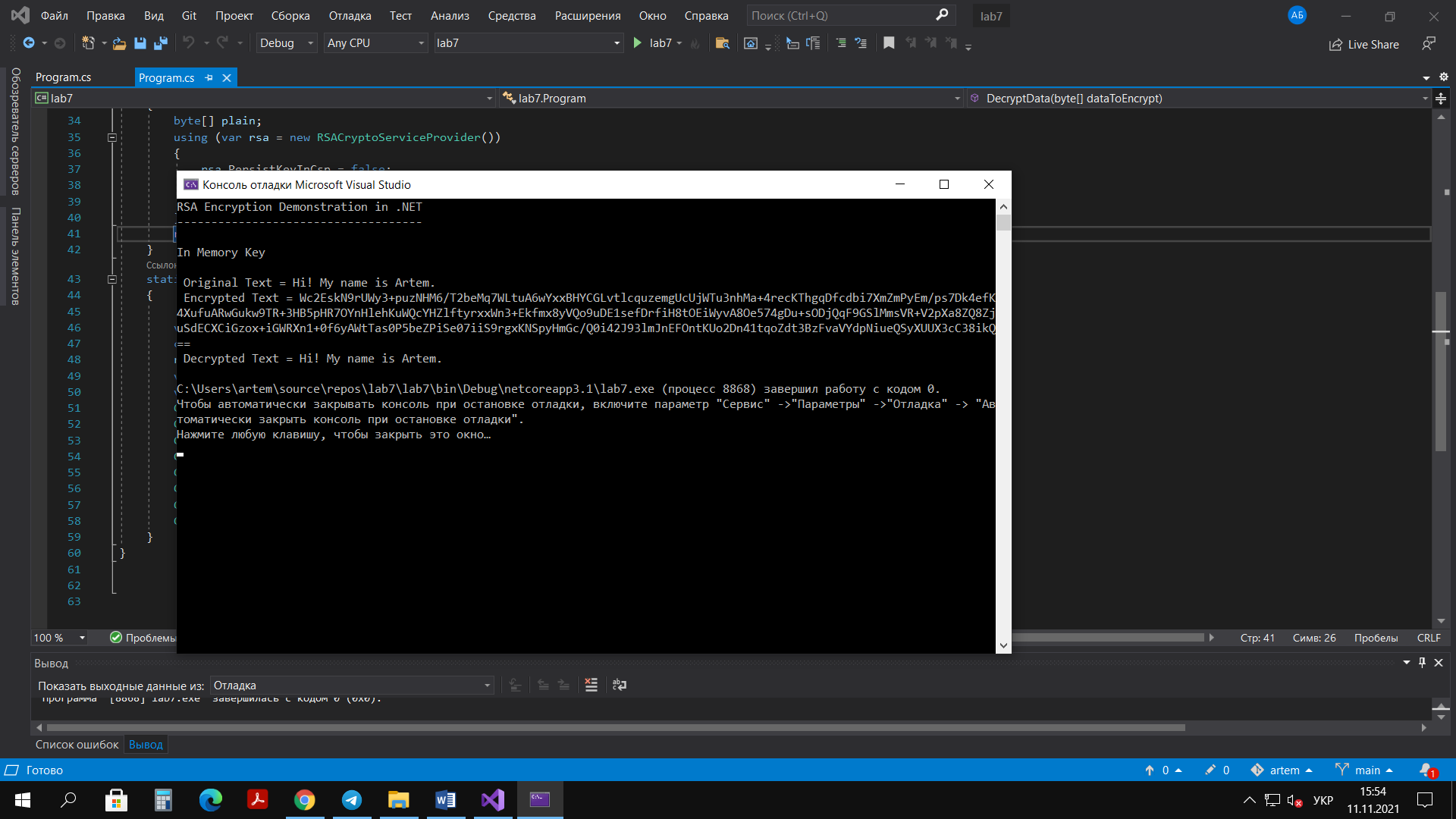
Переходимо до шифрування наших даних. Імпортуємо наш відкритий ключ для шифрування за допомогою rsa.ImportParameters(). Виконуємо шифрування rsa.Encrypt()



Далі потрібно розшифрувати повідомлення. Знову імпортуємо ключ, але в цей раз закритий, адже при асиметричному шифруванні для розшифрування використовуємо саме його. Виконуємо розшифрування rsa.Decrypt().



В кінці, ми повинні викликати метод, що викликає процес шифрування та дешифрування. Він виглядає таким чином, що зазначений на фото вище. Також задаємо повідомлення, яке хочемо зашифрувати та розшифрувати.



Результат виконання програми. Як можна бачити, що текст оригіналу та розшифрований повністю співпадають.

**Код:**

using System;

using System.Security.Cryptography;

using System.Text;

namespace lab7

{

class Program

{

private RSAParameters \_publicKey;

private RSAParameters \_privateKey;

public void AssignNewKey()

{

using(var rsa = new RSACryptoServiceProvider(2048))

{

rsa.PersistKeyInCsp = false;

\_publicKey = rsa.ExportParameters(false);

\_privateKey = rsa.ExportParameters(true);

}

}

public byte[] EncryptData(byte[] dataToEncrypt)

{

byte[] cipherbytes;

using (var rsa = new RSACryptoServiceProvider())

{

rsa.PersistKeyInCsp = false;

rsa.ImportParameters(\_publicKey);

cipherbytes = rsa.Encrypt(dataToEncrypt, true);

}

return cipherbytes;

}

public byte[] DecryptData(byte[] dataToEncrypt)

{

byte[] plain;

using (var rsa = new RSACryptoServiceProvider())

{

rsa.PersistKeyInCsp = false;

rsa.ImportParameters(\_privateKey);

plain = rsa.Decrypt(dataToEncrypt, true);

}

return plain;

}

static void Main(string[] args)

{

var rsaParams = new Program();

const string original = "Hi! My name is Artem.";

rsaParams.AssignNewKey();

var encrypted = rsaParams.EncryptData(Encoding.UTF8.GetBytes(original));

var decrypted = rsaParams.DecryptData(encrypted);

Console.WriteLine("RSA Encryption Demonstration in .NET");

Console.WriteLine("------------------------------------");

Console.WriteLine();

Console.WriteLine("In Memory Key");

Console.WriteLine();

Console.WriteLine(" Original Text = " + original);

Console.WriteLine(" Encrypted Text = " + Convert.ToBase64String(encrypted));

Console.WriteLine(" Decrypted Text = "+ Encoding.Default.GetString(decrypted));

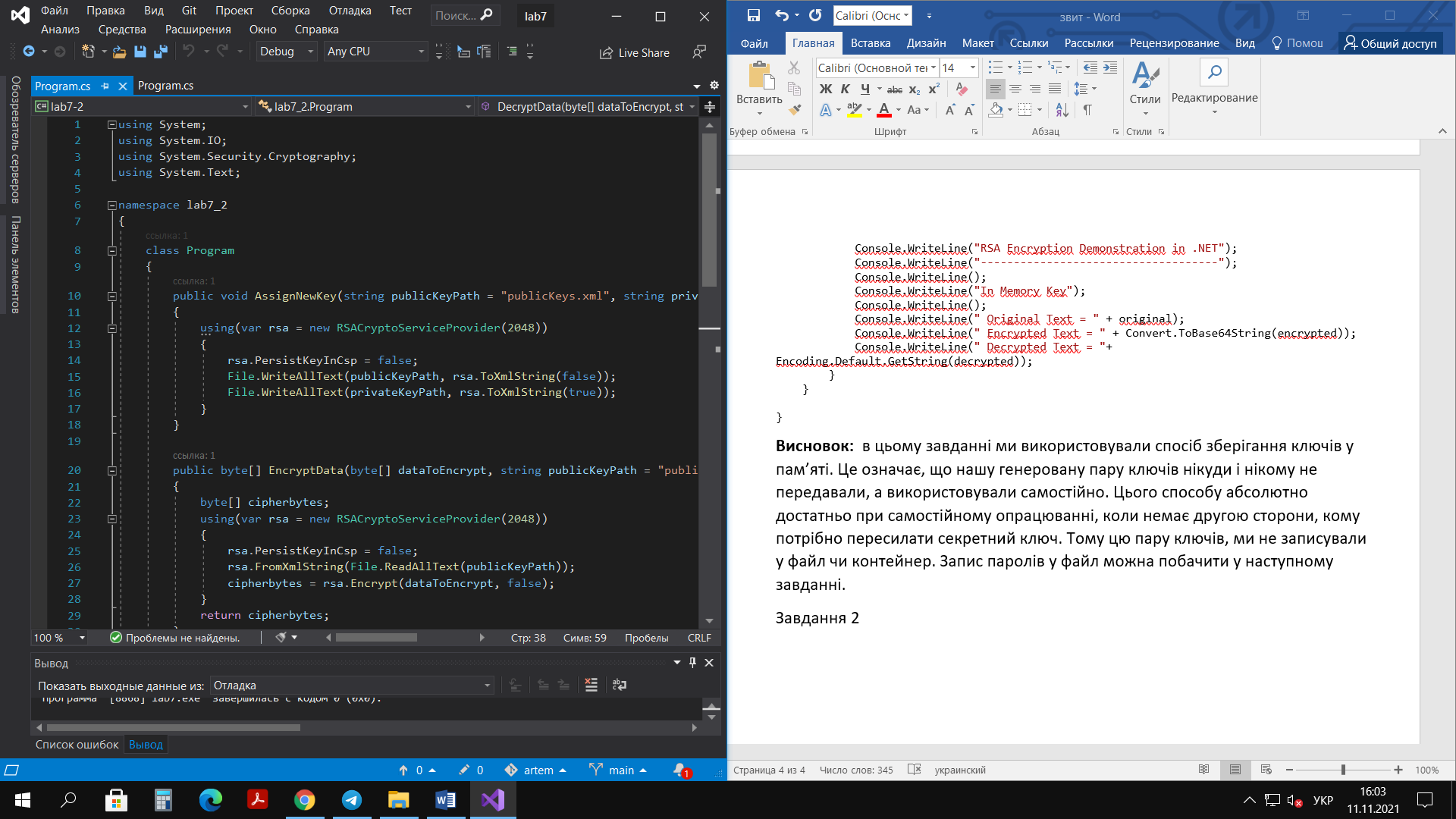
}

}

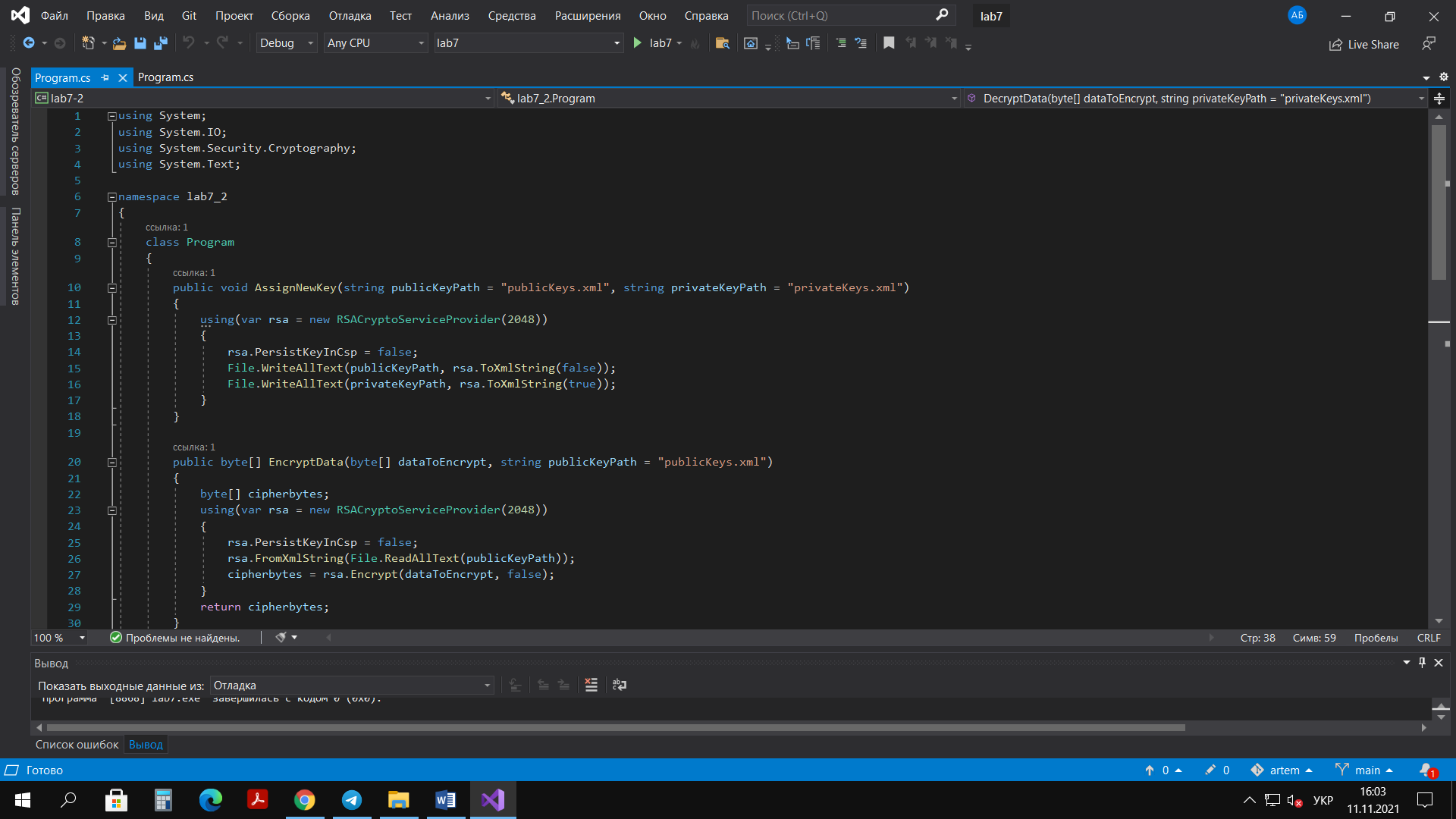
}

**Висновок:** в цьому завданні ми використовували спосіб зберігання ключів у пам’яті. Це означає, що нашу генеровану пару ключів нікуди і нікому не передавали, а використовували самостійно. Цього способу абсолютно достатньо при самостійному опрацюванні, коли немає другою сторони, кому потрібно пересилати секретний ключ. Тому цю пару ключів, ми не записували у файл чи контейнер. Запис паролів у файл можна побачити у наступному завданні.

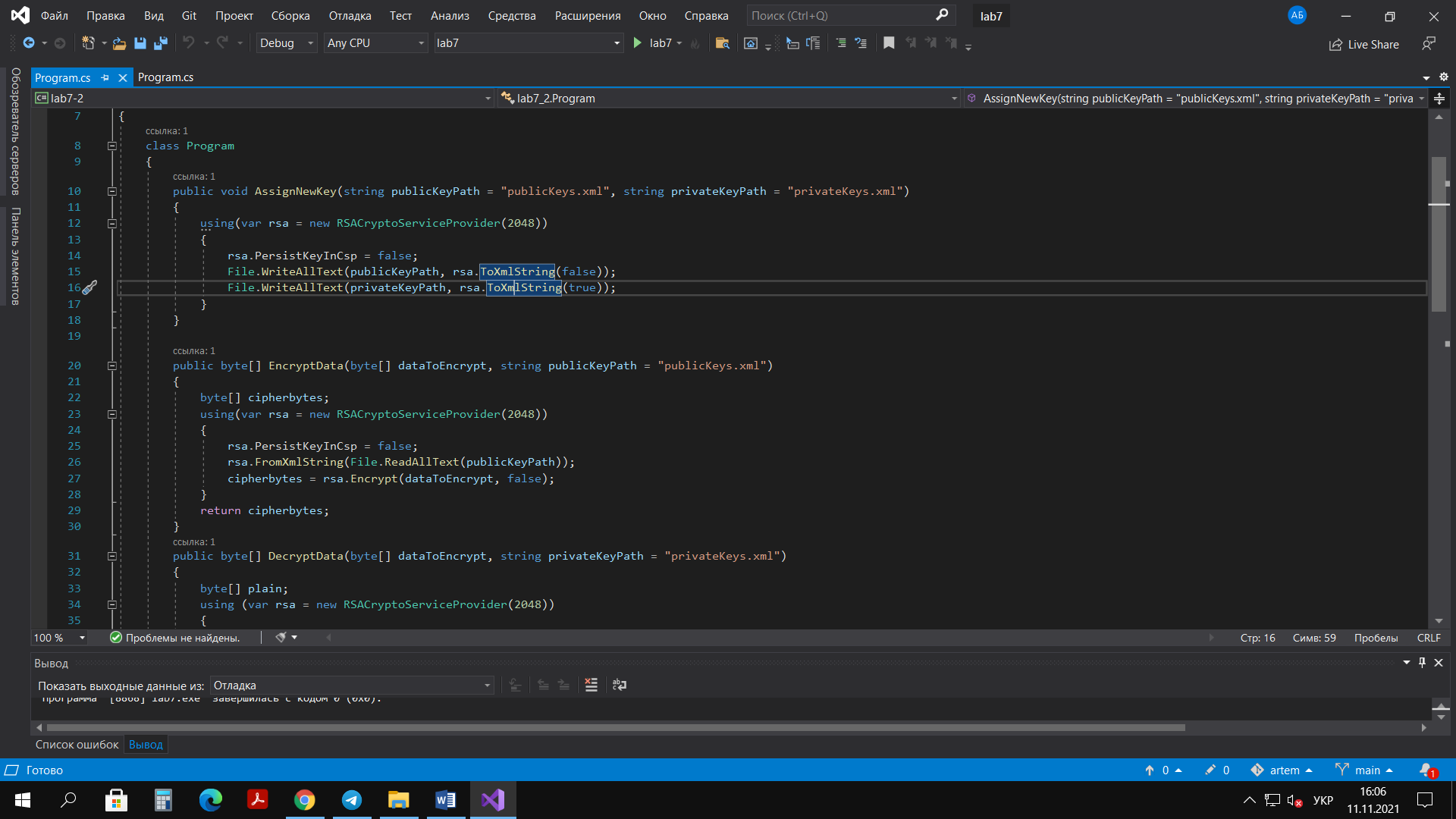
Завдання 2



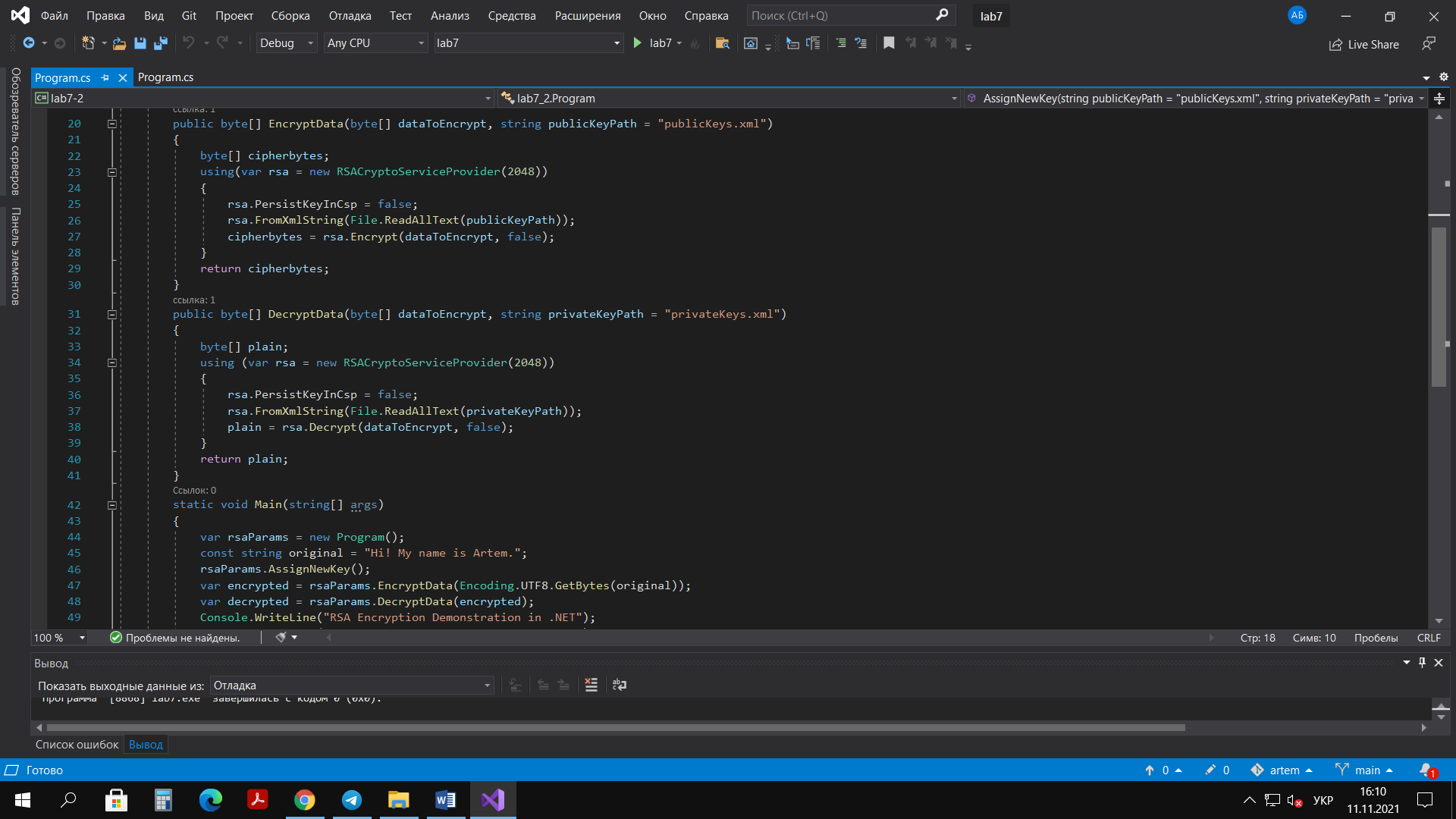
Використовує такі простори імен



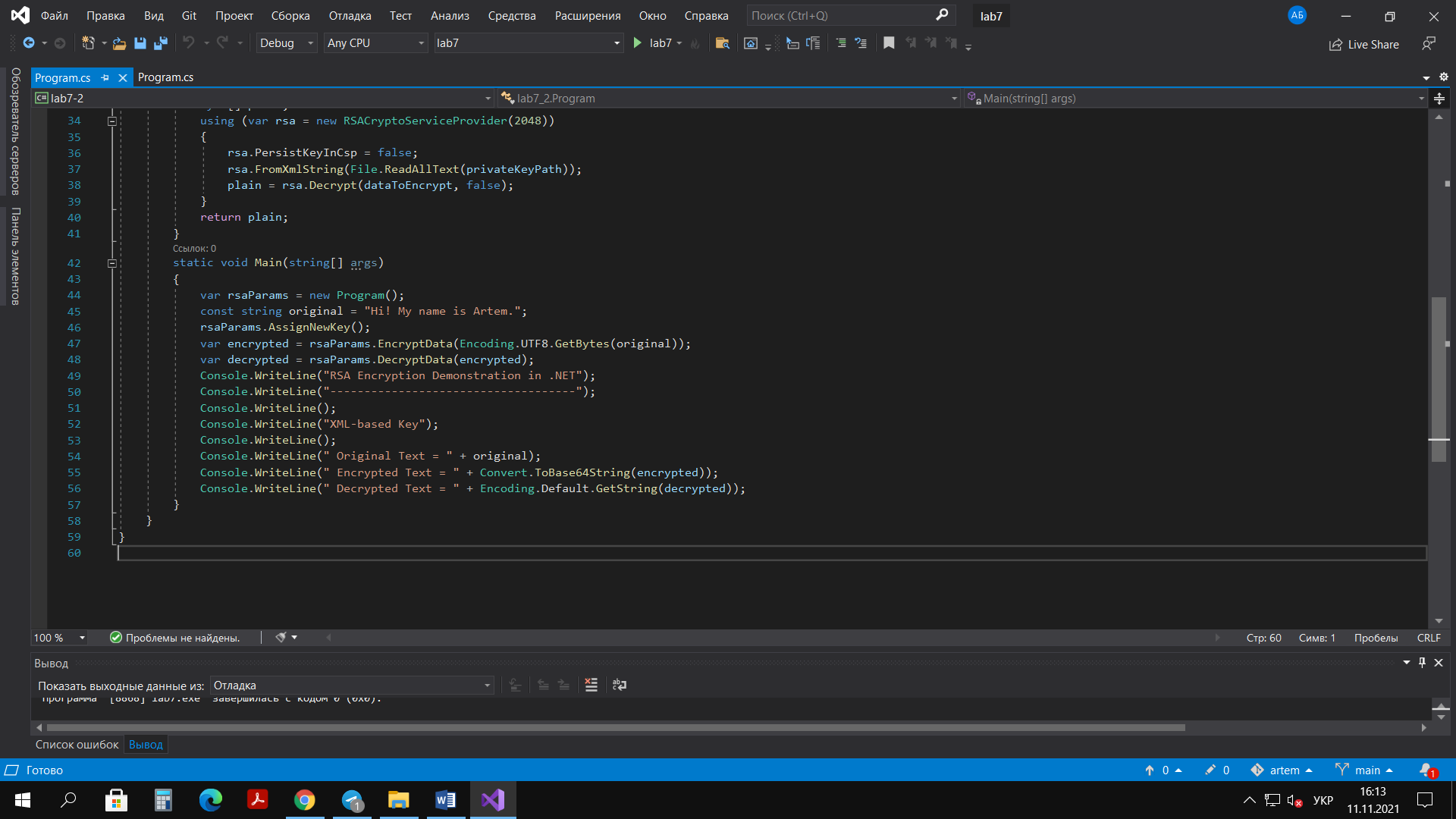
Створюємо два файли з розширенням .xml. Це файли для запису відритого та закритого ключа publicKeys.xml та privateKeys.xml відповідно. Та передаємо шлях до цих файлів у змінні. Генеруємо пару ключів та записуємо їх у відповідний файл за допомогою File.WriteAllText()



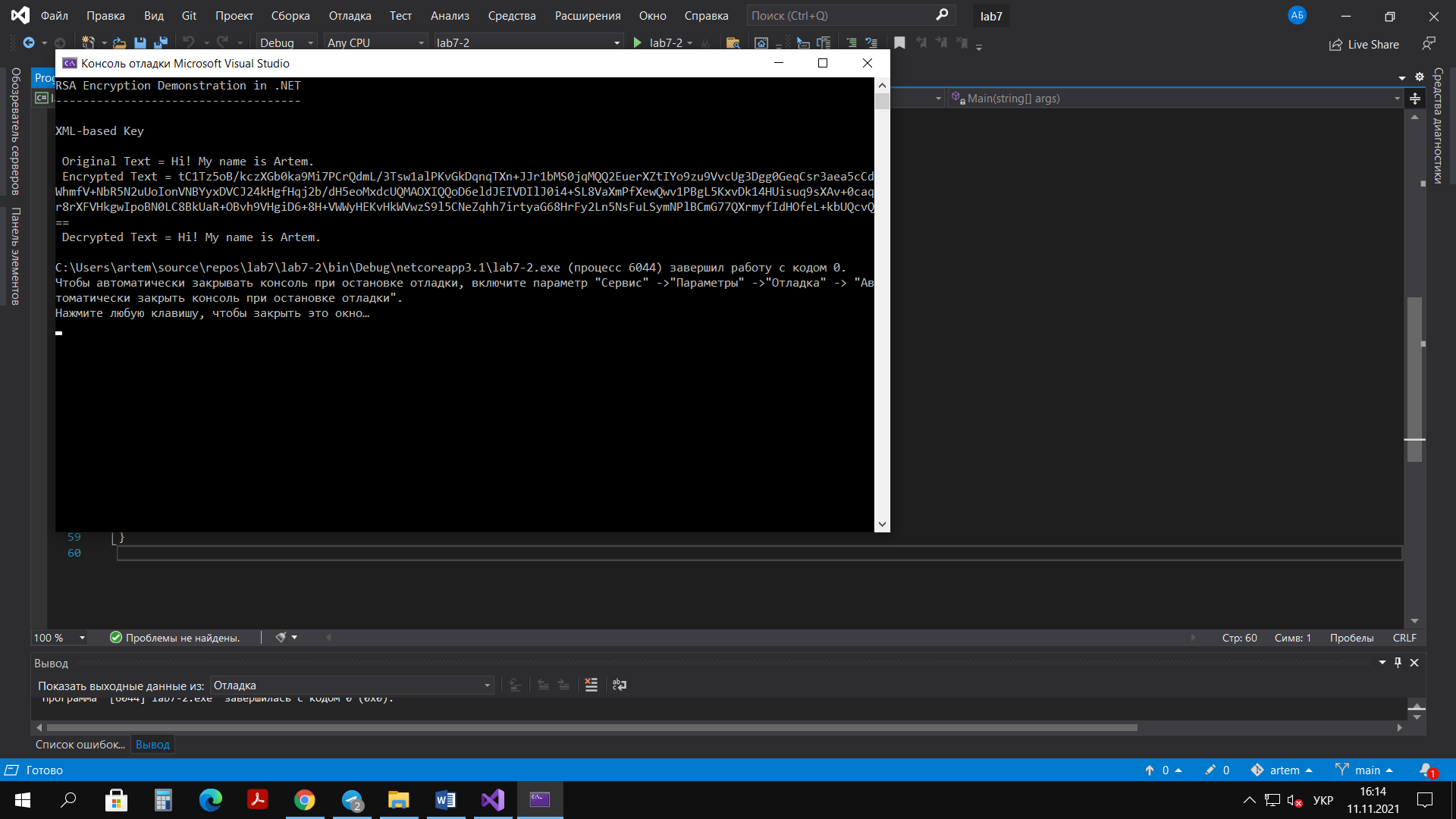
Далі переходимо до шифрування даних. По-перше зчитуємо наш відкритий ключ(він використовується при асиметричному шифруванні) за допомогою rsa.FromXmlString(File.ReadAllText()) та використовуємо його. Шифруємо за допомогою rsa.Encrypt().

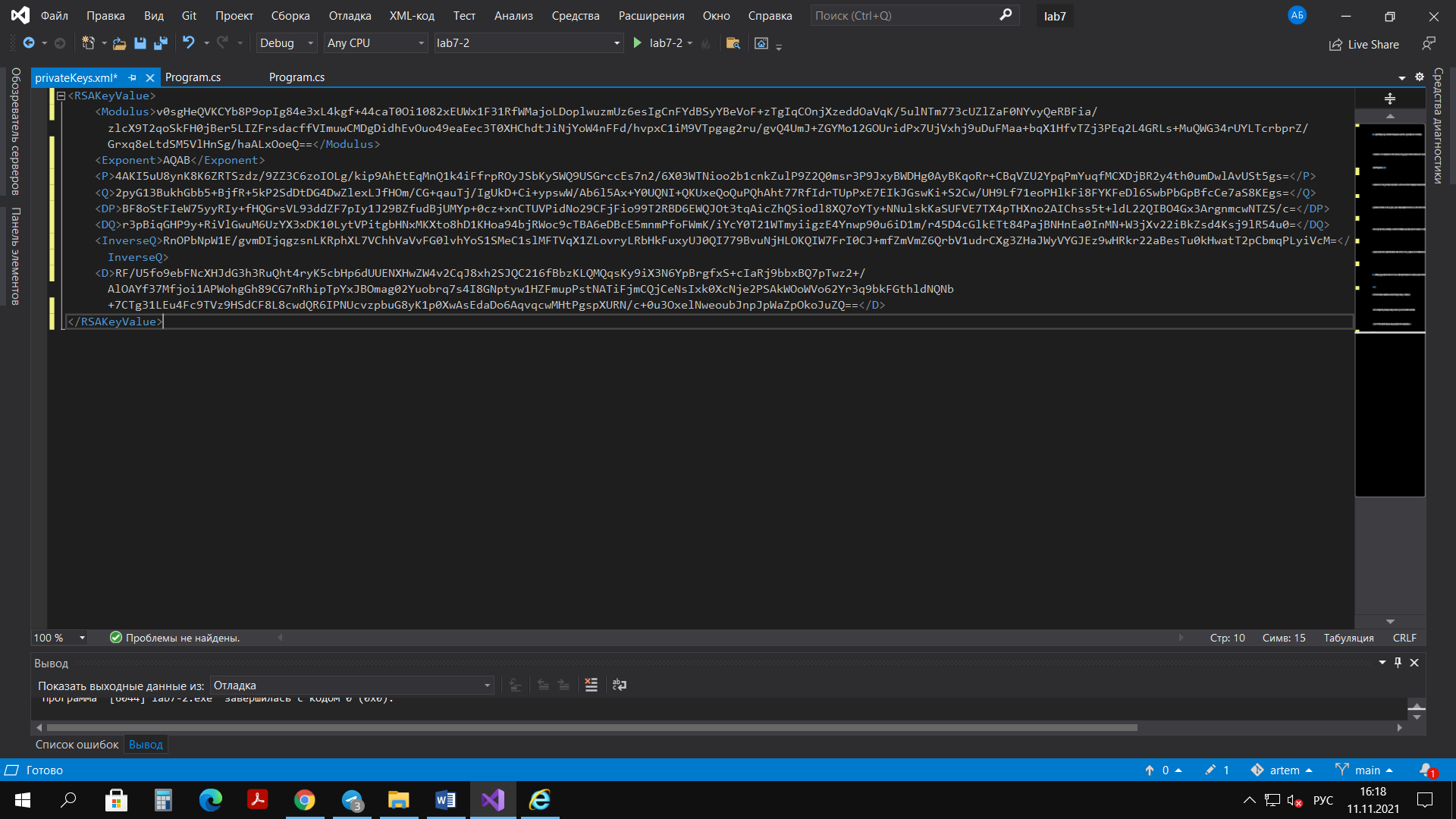


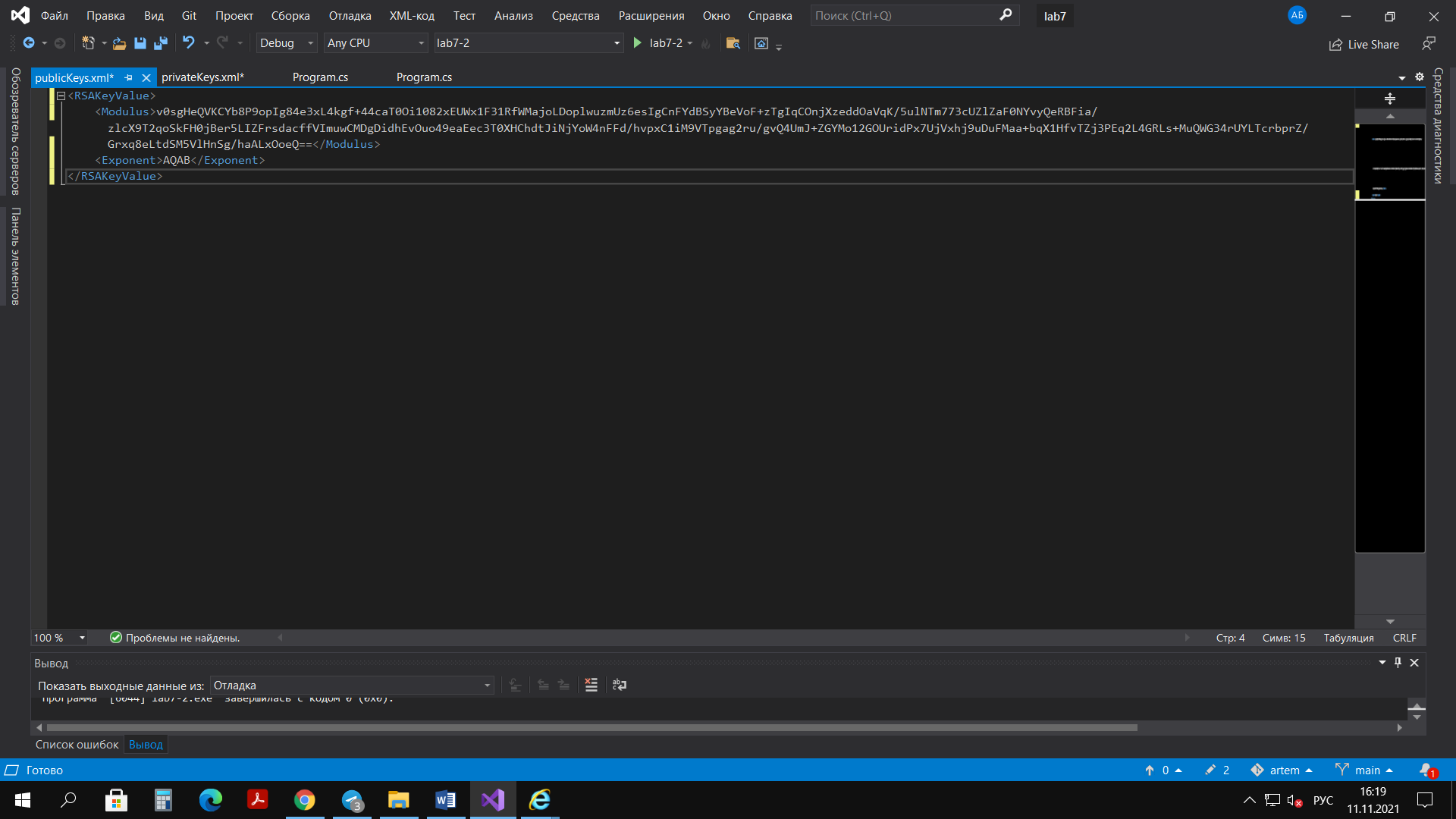
Після цього потрібно розшифрувати повідомлення. Так само зчитуємо наш ключ, але в цьому випадку використовуємо закритий ключ та використовуємо його. Розшифровуємо повідомлення rsa.Decrypt().



В кінці, ми повинні викликати метод, що викликає процес шифрування та дешифрування. Він виглядає таким чином, що зазначений на фото вище. Також задаємо повідомлення, яке хочемо зашифрувати та розшифрувати.







Результат виконання програми. Як можна бачити, що текст оригіналу та розшифрований повністю співпадають. Також можна побачити, що в результаті виконання програми до наших файлів записалися секретні та відкритий ключі.

**Код:**

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

namespace lab7\_2

{

class Program

{

public void AssignNewKey(string publicKeyPath = "publicKeys.xml", string privateKeyPath = "privateKeys.xml")

{

using(var rsa = new RSACryptoServiceProvider(2048))

{

rsa.PersistKeyInCsp = false;

File.WriteAllText(publicKeyPath, rsa.ToXmlString(false));

File.WriteAllText(privateKeyPath, rsa.ToXmlString(true));

}

}

public byte[] EncryptData(byte[] dataToEncrypt, string publicKeyPath = "publicKeys.xml")

{

byte[] cipherbytes;

using(var rsa = new RSACryptoServiceProvider(2048))

{

rsa.PersistKeyInCsp = false;

rsa.FromXmlString(File.ReadAllText(publicKeyPath));

cipherbytes = rsa.Encrypt(dataToEncrypt, false);

}

return cipherbytes;

}

public byte[] DecryptData(byte[] dataToEncrypt, string privateKeyPath = "privateKeys.xml")

{

byte[] plain;

using (var rsa = new RSACryptoServiceProvider(2048))

{

rsa.PersistKeyInCsp = false;

rsa.FromXmlString(File.ReadAllText(privateKeyPath));

plain = rsa.Decrypt(dataToEncrypt, false);

}

return plain;

}

static void Main(string[] args)

{

var rsaParams = new Program();

const string original = "Hi! My name is Artem.";

rsaParams.AssignNewKey();

var encrypted = rsaParams.EncryptData(Encoding.UTF8.GetBytes(original));

var decrypted = rsaParams.DecryptData(encrypted);

Console.WriteLine("RSA Encryption Demonstration in .NET");

Console.WriteLine("------------------------------------");

Console.WriteLine();

Console.WriteLine("XML-based Key");

Console.WriteLine();

Console.WriteLine(" Original Text = " + original);

Console.WriteLine(" Encrypted Text = " + Convert.ToBase64String(encrypted));

Console.WriteLine(" Decrypted Text = " + Encoding.Default.GetString(decrypted));

}

}

}

**Висновок:** під час цього завдання ми записували пару ключів у файл та потім використовували їх для шифрування та розшифрування даних. Такий спосіб є вкрай ненадійним, адже весь сенс в асиметричному шифруванні є в тому, що ми використовуємо два ключі. Одним ключем ми шифруємо – відкритий ключ, другим ключем інша особам, яка його має, розшифровує – закритий ключ. Відкритий ключ може знати будь-хто, але закритий відомий лише отримувачу. При відправці файлу з секретним ключем зловмисник має змогу просто перехопити сам файл та прочитати секретний ключ. Без необхідності самому цей ключ підбирати, він використає перехоплений ключ. Саме тому цей спосіб не є надійним в плані збереження ключів