***Практичне заняття №7-8***

*«*Асиметричне шифрування як засіб забезпечення конфіденційності інформації*»*

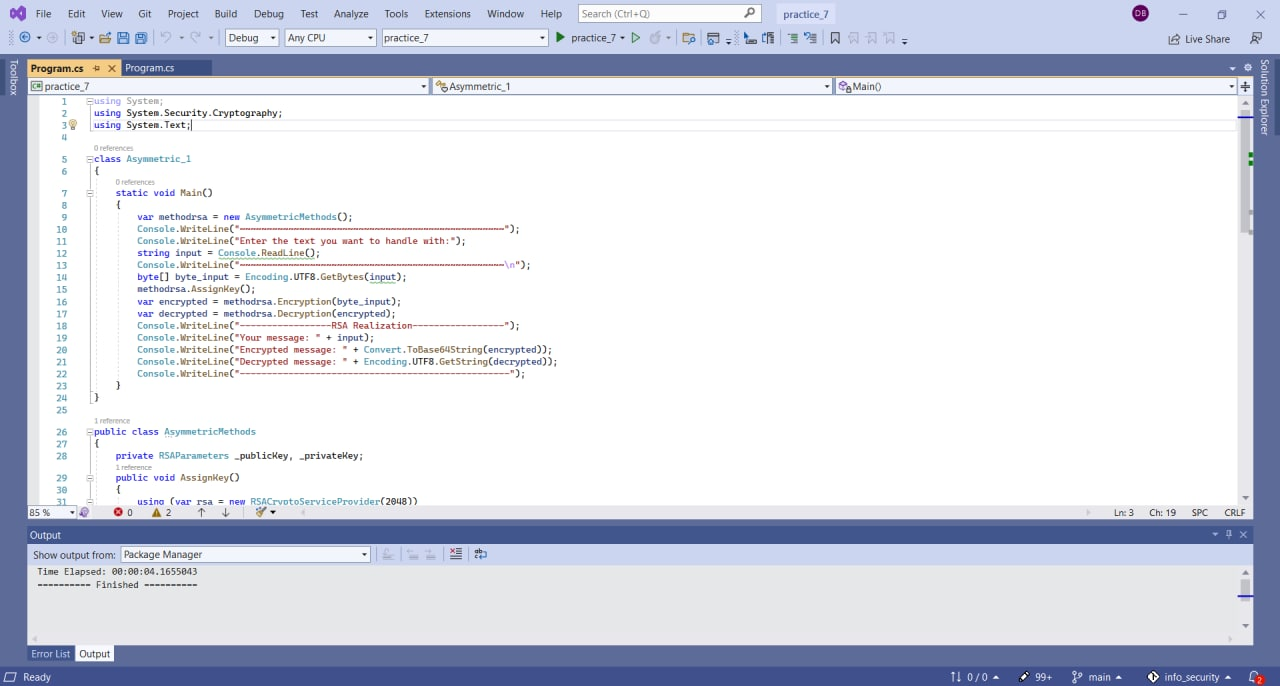
Виконала студентка групи МІТ-21 **Борук Дарина**

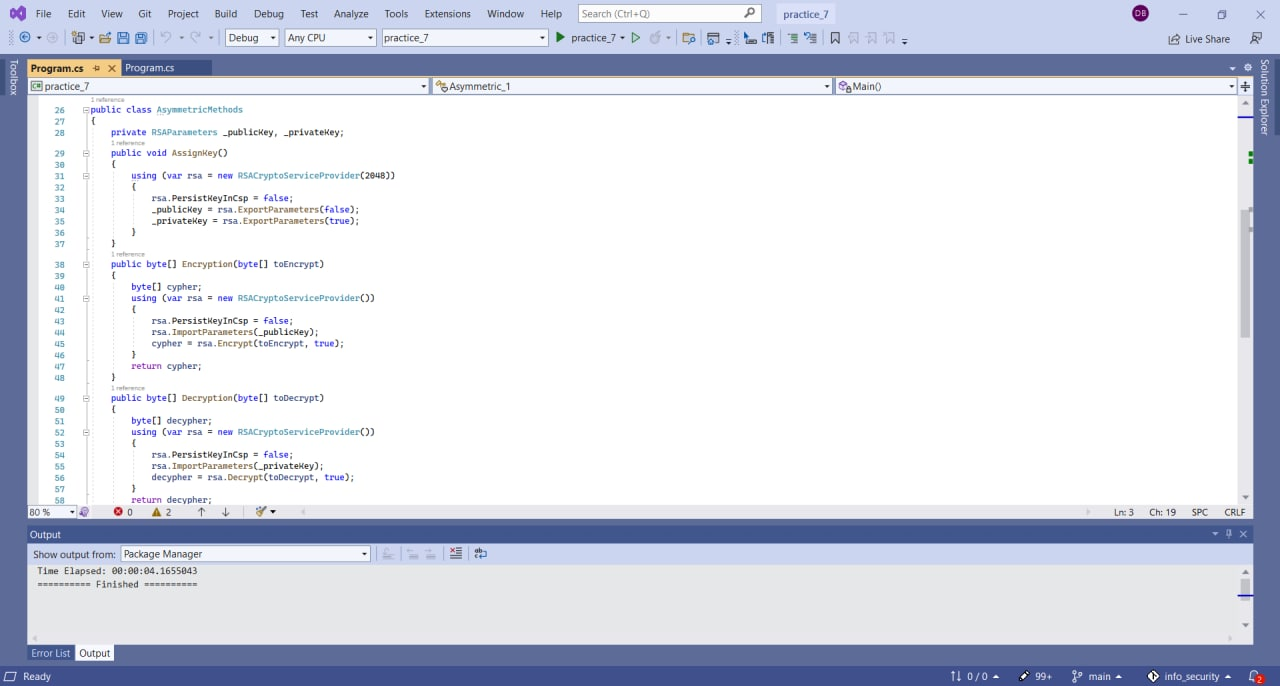
Мета: розібратися із асиметричним шифруванням, використанням криптографічних бібліотек для його здійснення і варіантами збереження відкритого ключа та його зчитування. Написати програму, яка виконує зашифровування та розшифровування даних з використанням алгоритмів асиметричного шифрування RSA, пара ключів зберігається у пам'яті. Також реалізувати можливість збереження відкритого ключа у файлі із можливістю його подальшого використання для зашифровування

Хід роботи:

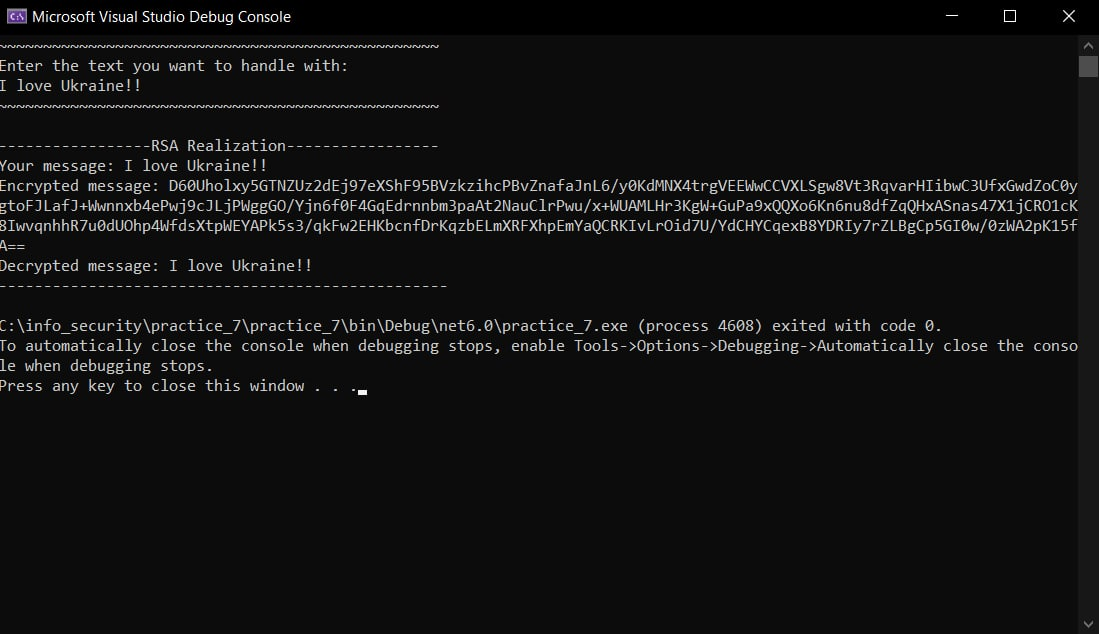
* Розпочнемо із знайомства з асиметричним шифруванням і напишемо загальну програму, яка використовує алгоритми асиметричного шифрування для зашифровування і розшифровування повідомлення.

1. Для початку пропишемо головну функцію, де введене повідомлення користувача буде передано у функції шифрування і розшифровування.
2. Створимо клас AsymmetricMethods, в якому пропишемо наші загальні функції. Пропишемо функцію, яка згенерує нам пару ключів: відкритий і секретний, які будуть зберігатись у пам'яті. Щоб експортувати інформацію для наших ключів, ми викличемо з об'єкта RSA екземпляр RSACryptoServiceProvider і метод ExportParameters. Щоб експортувати публічний ключ ми передаємо значення false, а щоб експортувати секретний ключ – значення true. Властивість PersistKeyInCsp використовується для збереження ключів у контейнері, так як ми їх не зберігаємо у контейнері, даємо цій властивості значення false. Далі перейдемо безпосередньо до написання функцій шифрування і розшифровування. Ці функції схожі між собою, єдина різниця у виклику ключа для реалізації: для шифрування використовується публічний ключ, а для розшифрування – секретний. В обох випадках нашими параметрами є дані, які ми передаємо у вигляді масиву байтів і хочемо зашифрувати (розшифрувати). Далі створюємо змінну для запису нашого результату і, створюючи новий об'єкт RSACryptoServiceProvider, імпортуємо публічний ключ (секретний ключ) за допомогою методу ImportParamters і ставимо значення false для нашої властивості PersistKeyInCsp. Далі викличемо метод Encrypt (Decrypt), в якому першим параметром будемо передавати наші дані для роботи, а другим – значення, що є оптимальним доповненням асиметричного шифрування і яке додає більшу криптостійкість до нашого процесу, дамо йому значення true. І в кінці повертаємо масив байтів нашого зашифрованого (розшифрованого) повідомлення.





1. Маємо приклад виконання програми:



* Далі для попередньої програми реалізуємо можливість збереження відкритого ключа у файл з назвою, що відповідає транслітерації прізвища, щоб потім їм можна було поділитись і зашифрувати повідомлення використовуючи цей файл.

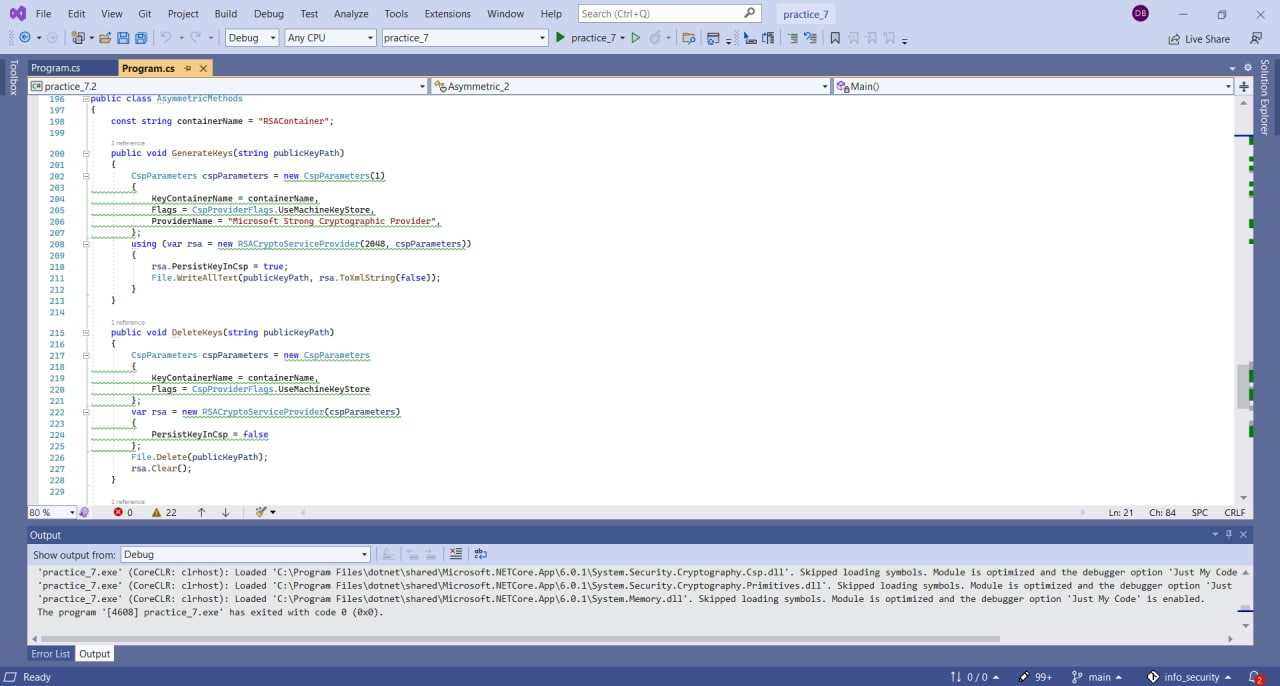
1. У головній функції пропишемо змінні, які будуть використовуватись у нашій програмі: змінна для нашого шляху для збереження публічних ключів і дві змінні для шляху, в якому будуть зберігатись зашифровані файли. Також пропишемо змінні для запису прізвища і подальшого його використання і змінну для створення певного нового шляху. Пропишемо меню для користувача з вільним вибором опції, що саме він хоче зробити: створити пару ключів і зберегти публічний у файл, видалити ключі, зашифрувати повідомлення, розшифрувати повідомлення.
2. Розглянемо клас AsymmetricMethods, в якому пропишемо усі наші функції для реалізації наших опцій. Першою такою функцією є функція генерації пари ключів. Для збереження відкритого ключа у файлі нам потрібно створити контейнер, в якому буде зберігатись секретний ключ. Ми створюємо змінну, яка буде нашим контейнером і даємо їй певну назву. Для роботи з контейнером нам спочатку треба створити екземпляр класу CspParameters і надати йому назву нашого контейнера у поле CspParameters.KeyContainerName. Далі використовуючи властивість Flags, ми можемо обрати як саме буде зберігатись ключ – локально у користувацькому контейнері чи на рівні машини. У нашому випадку ми зберігаємо на рівні машини. Назвою контейнера за замовчуванням у Windows є Microsoft Strong Cryptographic Provider. Цей контейнер нам знадобиться для розшифровування зашифрованого повідомлення, так як сааме у ньому буде зберігатись секретний ключ, який, як нам відомо, допоможе нам розшифрувати повідомлення. Далі ми створюємо об'єкт екземпляру RSACryptoServiceProvider і передамо туди значення довжини нашого ключа (2048) і створений екземпляр класу CspParameters, який буде використовуватись для необхідної інформації. У цьому об'єкті ми надаємо властивості PersistKeyInCsp значення true (означає, що ключ ми зберігаємо у контейнері) і за допомогою методів роботи з файлами створюємо і одразу ж записуємо значення нашого публічного ключа у файл зі шляхом, який буде передаватись параметром нашої функції.

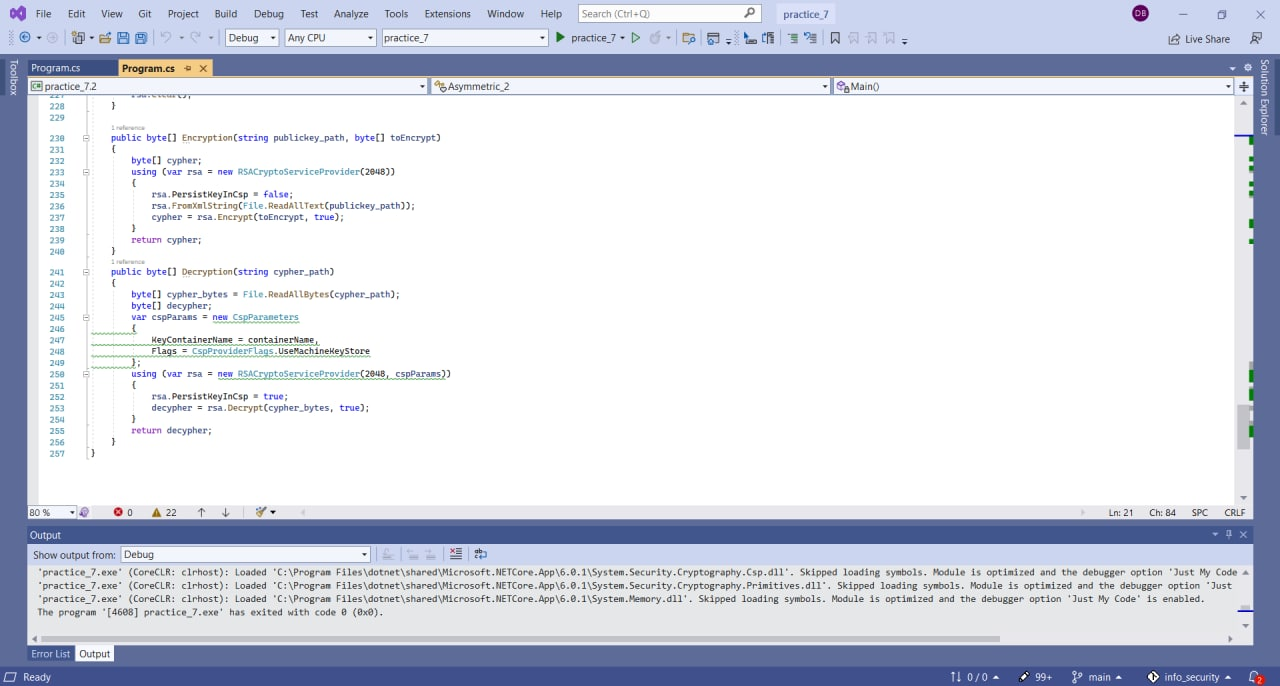
Для того, щоб видалити ключі ми створюємо той самий екземпляр класу CspParameters і прописуємо значення false властивості PersistKeyInCsp. Далі просто видаляємо файл за допомогою метода File.Delete(наш шлях) і очищуємо контейнер за допомогою метода Clear().

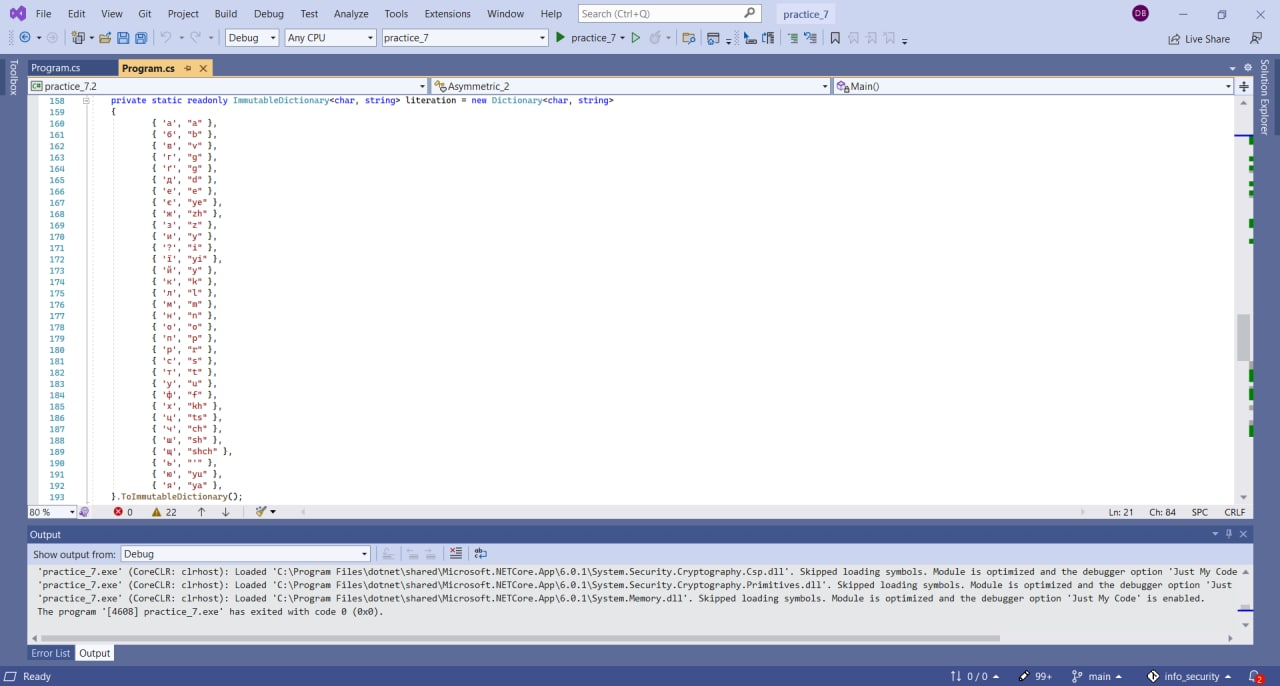
Далі у нас функція шифрування. Різницею цієї функції від функції у попередньому пункті є те, що у параметри ми ще передаємо шлях, з якого нам треба буде зчитати ключ для шифрування.

У функції дешифрування та ж сама ситуація: ми передаємо шлях, який веде нас до нашого зашифрованого повідомлення, з якого ми маємо зчитати його і розшифрувати. Значення PersistKeyInCsp у цій функції true, так як нам потрібно знати секретний ключ для розшифровування.

Тепер перейдемо до розбору самої програми.





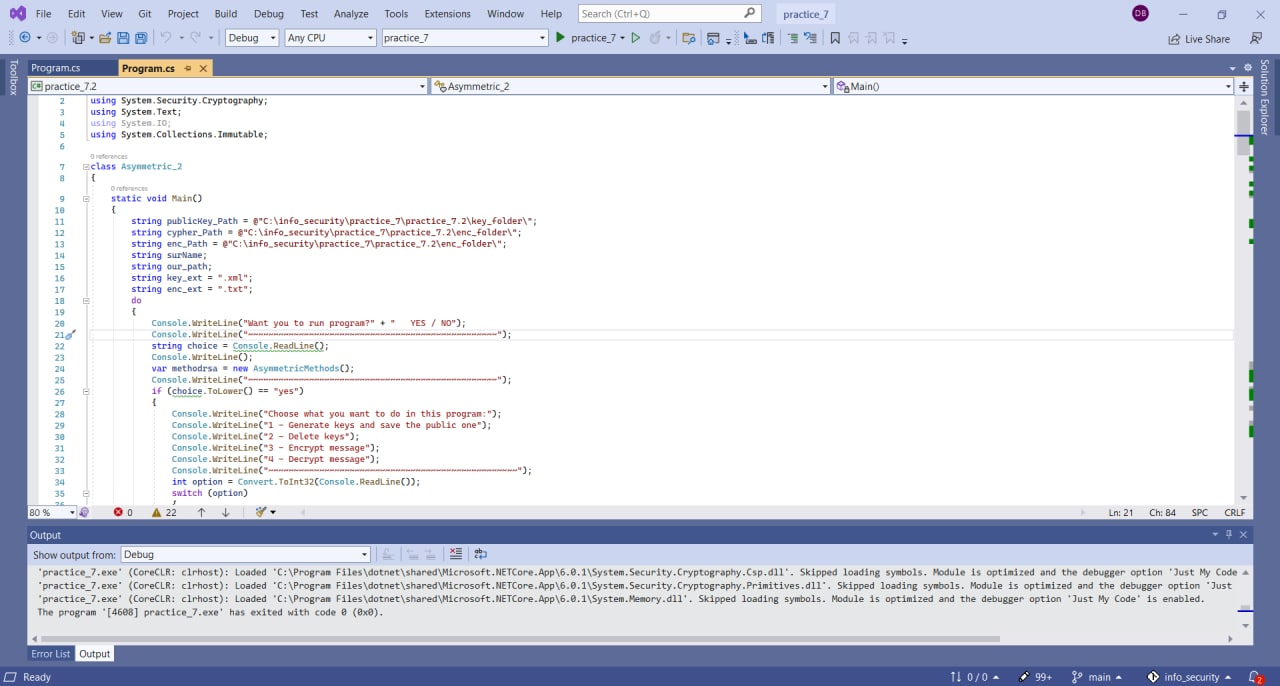


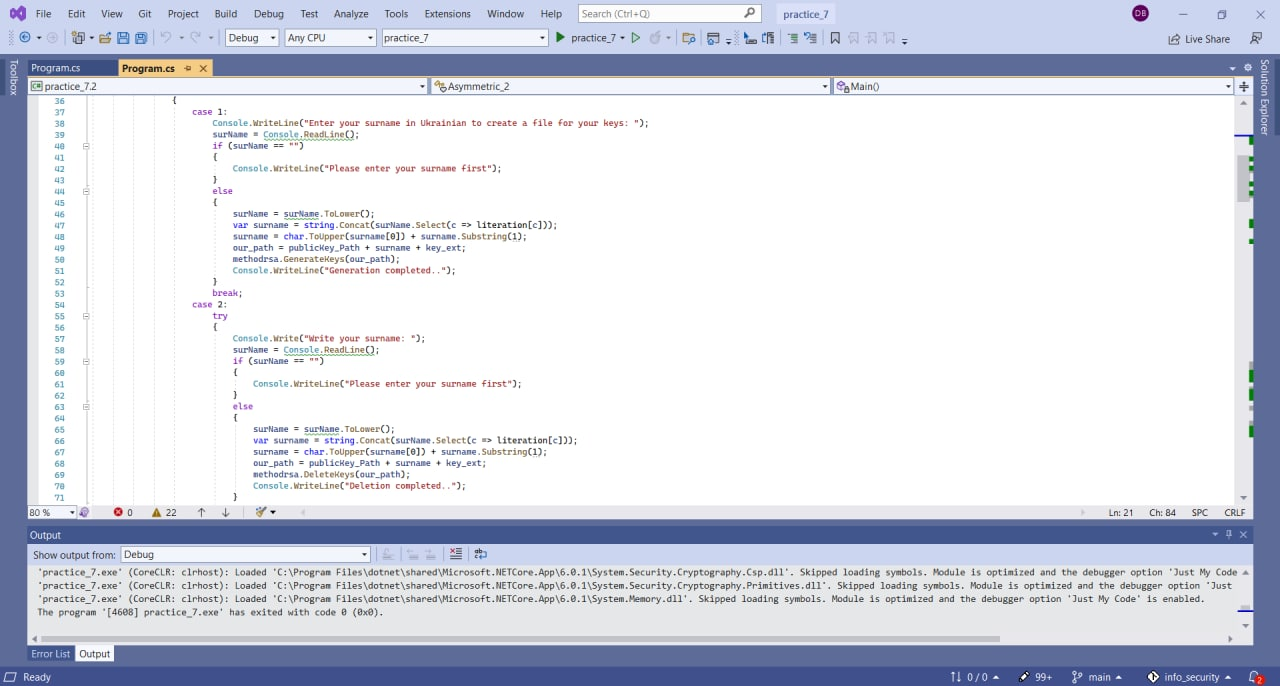
1. У випадку обрання першої опції користувач вводить своє прізвище українською мовою для назви файлу, де буде збережено публічний ключ, і яке буде підлягати транслітерації. Для транслітерації ми створимо колекцію ImmutableDictionary ключів і значень з пакету System.Collections.Immutable, в якій пропишемо латиницю і кирилицю, на яку вона має бути замінена. Далі за допомогою метода string.Concat поєднаємо наші значення в одне, яке і буде назвою нашого файлу. Далі в змінній нового шляху створимо шлях до нашого файлу: шляху для збереження публічних ключів + транслітероване прізвище + розширення txt. Далі цей шлях ми передаємо у функцію для створення ключів.

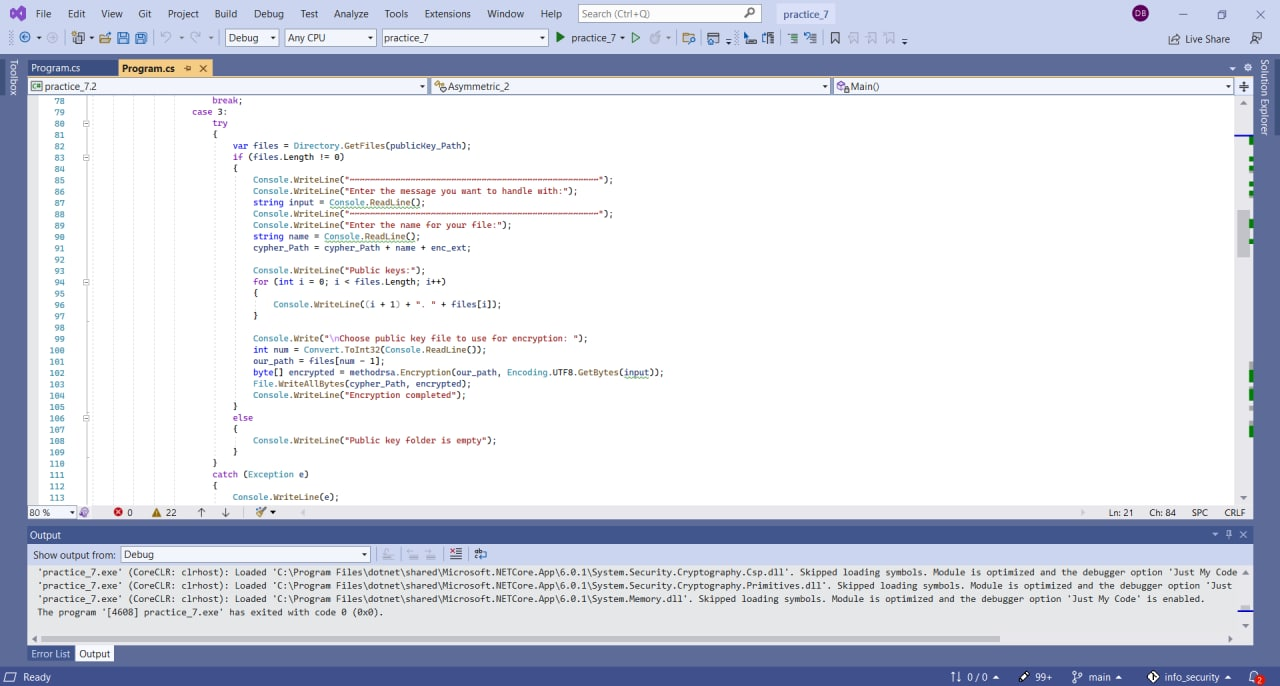
Для видалення ключа користувач вводить своє прізвище, програма його транслітерує і створює шлях до файлу з такою назвою. далі цей шлях передається у функцію видалення ключів.

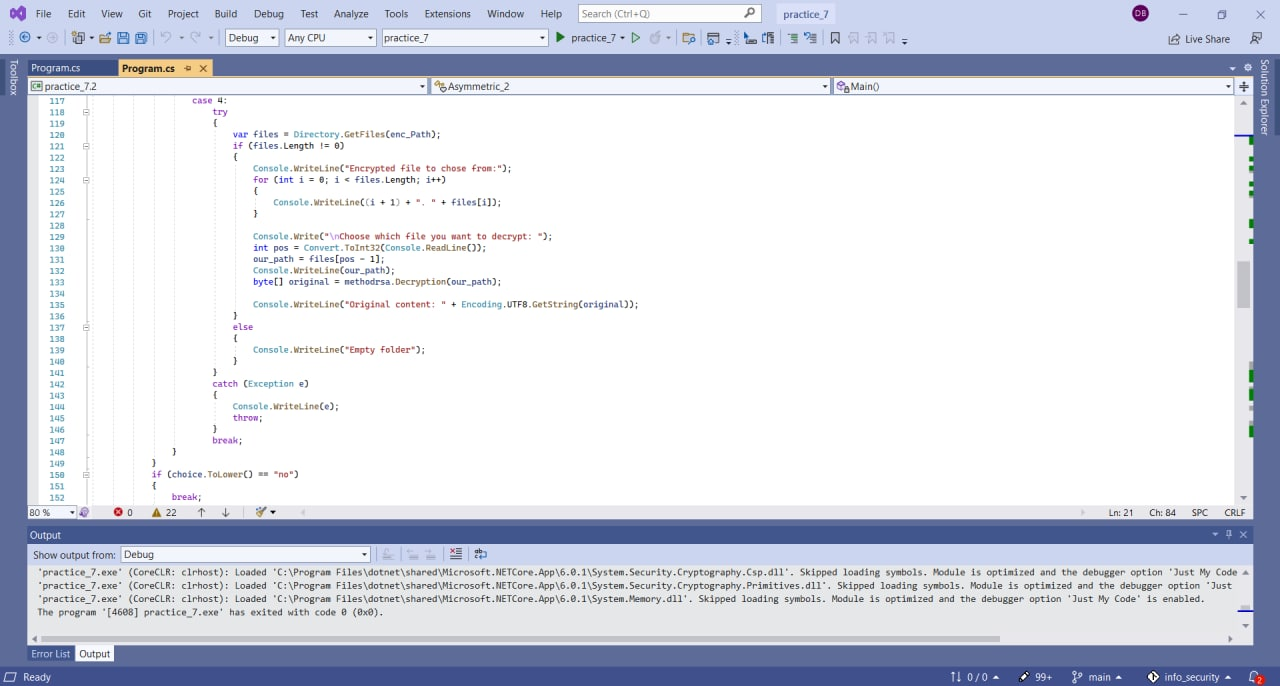
Для шифрування файлу нам потрібно обрати публічний ключ, яким ми будемо шифрувати, тому ми спочатку заходимо у папку з нашими ключами, використовуючи метод Directory.Getfiles(шлях до збережених ключів). Якщо папка не пуста, користувач вводить повідомлення, що хоче зашифрувати і вводить назву файлу, у який буде збережено зашифроване повідомлення. Далі створюється відповідний шлях до цього файлу. Потім користувач обирає, яким ключем він хоче зашифрувати своє повідомлення через вивід умісту папки. Далі створюємо змінну, у яку буде записано наше зашифроване повідомлення, через функцію шифрування, у яку ми передамо шлях до ключа, який обере користувач, і повідомлення, яке потрібно зашифрувати. Після виконання цього процесу викличемо метод роботи з файлами, який створить і запише результат процесу у шлях. який був створений на початку з назвою, введеною користувачем.

Для розшифровування буде схожий процес. Спочатку ми виводимо на екран файли (за допомогою цієї команди Directory.Getfiles()), які можна розшифрувати і користувач обирає, з яким саме він хоче працювати. Далі обраний файл, а точніше його шлях, передається у функцію розшифровування і потім вже результат функції ми конвертуємо у звичний нам вигляд рядка.



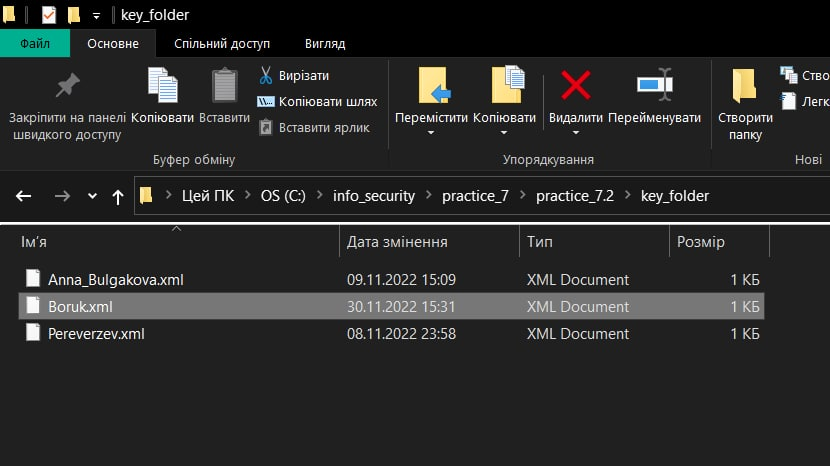
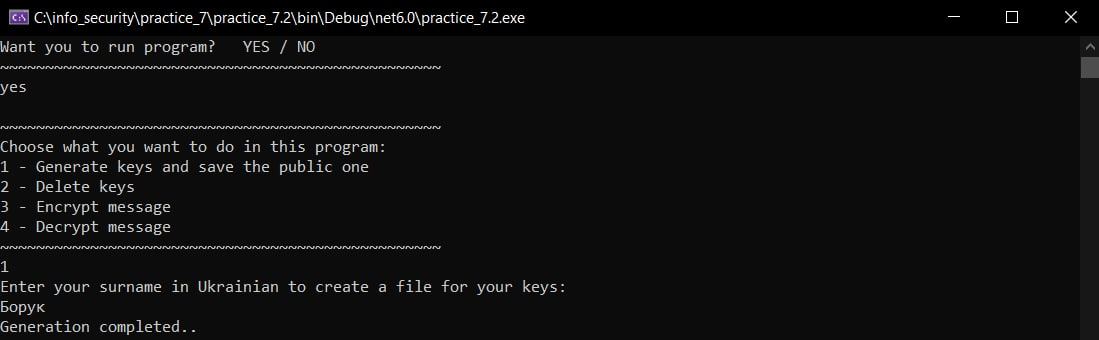






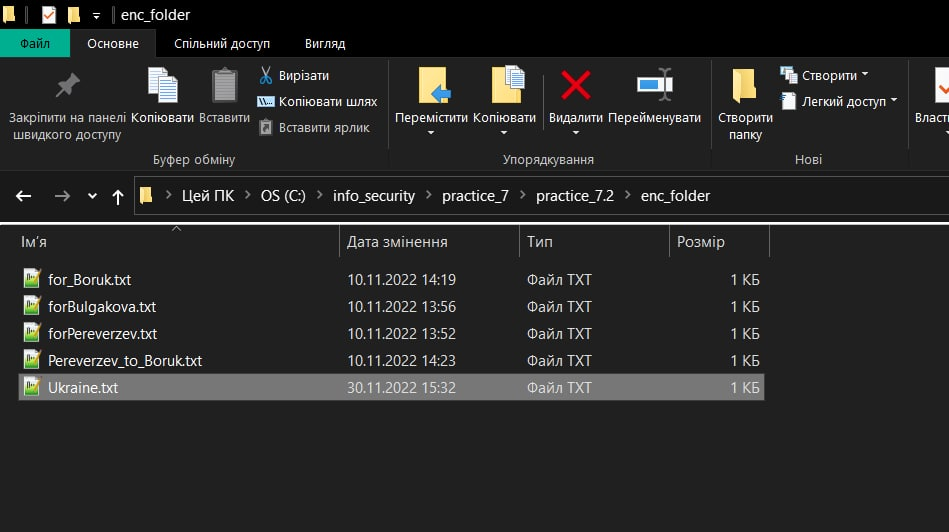
1. Приклад виконання програми:

* Для початку згенеруємо наші ключі

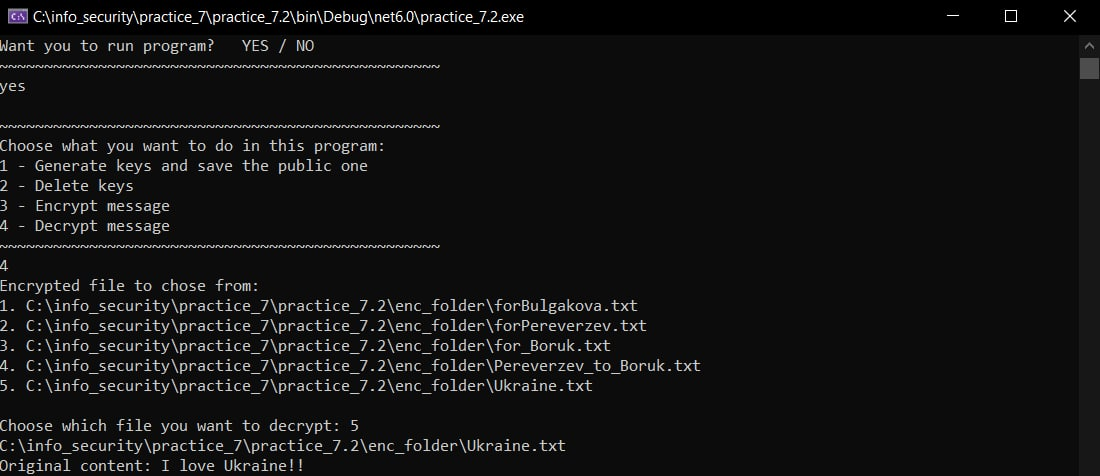


Можемо побачити, що наш ключ зберігся у папку з назвою мого прізвища (вже транслітерованого)

* Далі зашифруємо якесь повідомлення, використовуючи наш новостворений ключ

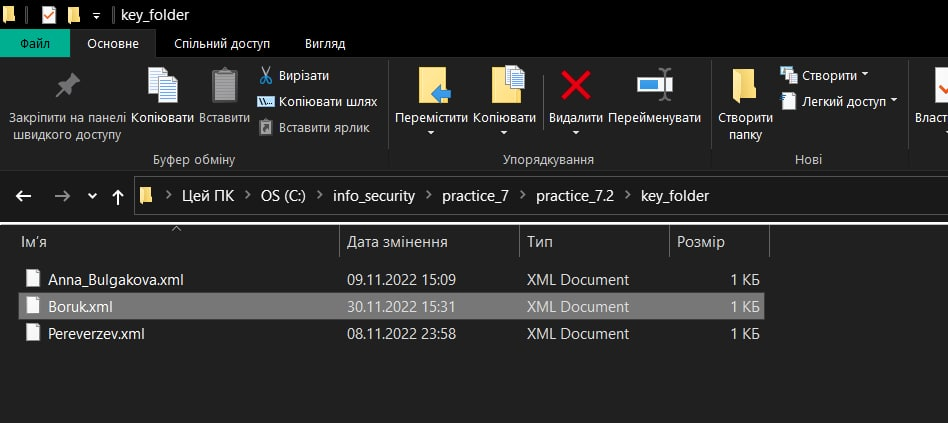


* І спробуємо тепер розшифрувати наше зашифроване повідомлення

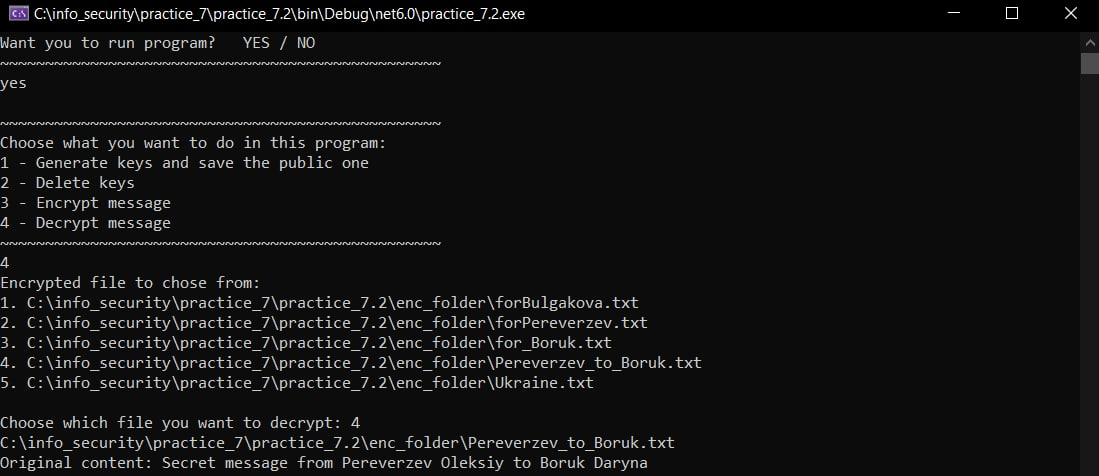
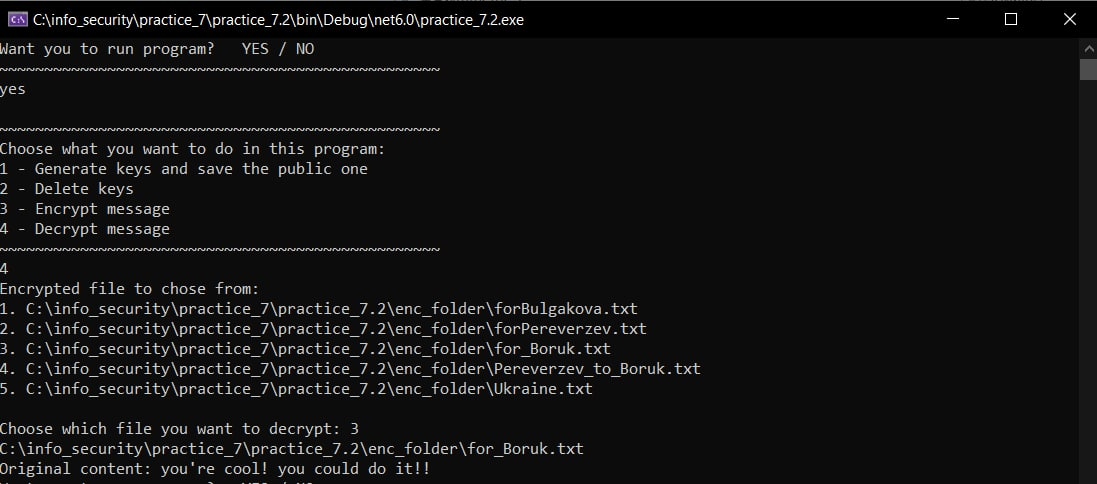


* Спробуємо тепер розшифрувати повідомлення, яке мені надіслали, використовуючи для шифрування мій публічний ключ.

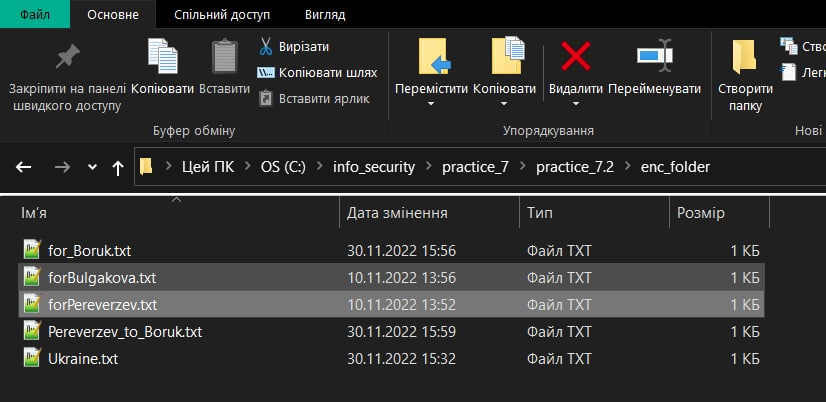
У мене є мій створений публічний ключ і 2 повідомлення, які мені треба розшифрувати, які я зберегла у папку



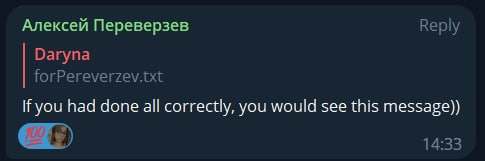
Використовуючи свій публічний ключ для розшифровування повідомлень, маємо:



* Так само і я зашифрувала 2 повідомлення, використовуючи завантажені публічні ключі, які іншим потрібно буде в себе розшифрувати.



Маємо результат:

Висновки: у ході практичної роботи були отримані знання про асиметричне шифрування, публічні і секретні ключі, їх варіанти зберігання і виклику, алгоритм асиметричного шифрування RSA.