***Практичне заняття №6***

*«Забезпечення конфіденційності інформації з використанням симетричного шифрування»*

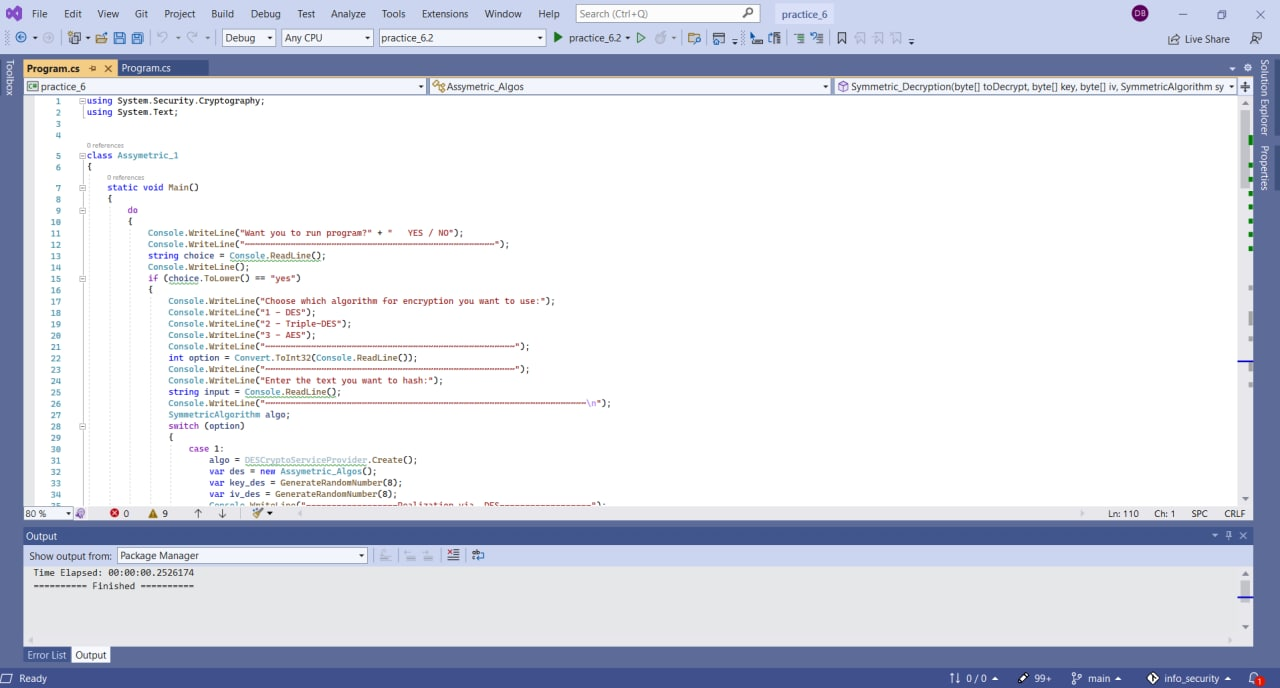
Виконала студентка групи МІТ-21 **Борук Дарина**

Мета: розібратися із системою симетричного шифрування, дізнатись про алгоритми DES, Triple – DES і AES.

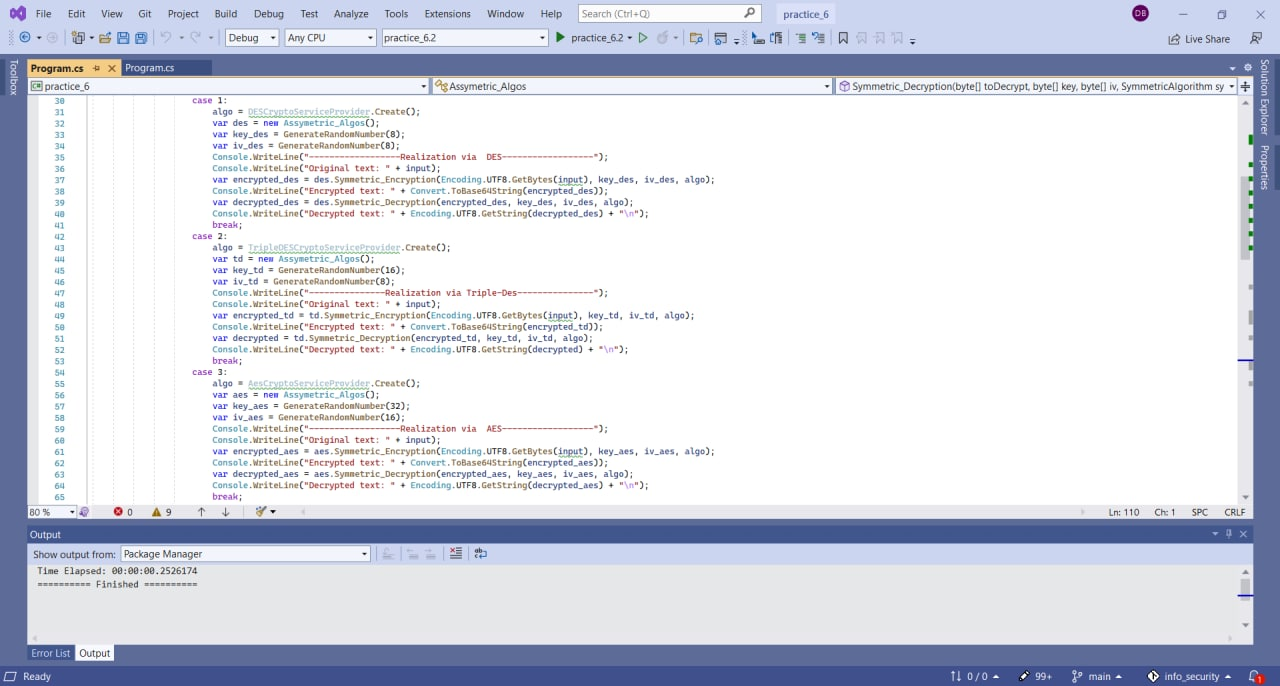
Хід роботи:

* Напишемо програму, яка виконує зашифровування та розшифровування даних з використанням алгоритмів симетричного шифрування DES, Triple – DES, AES. Секретний ключ та вектор ініціалізації буде генеруватись випадковим чином.

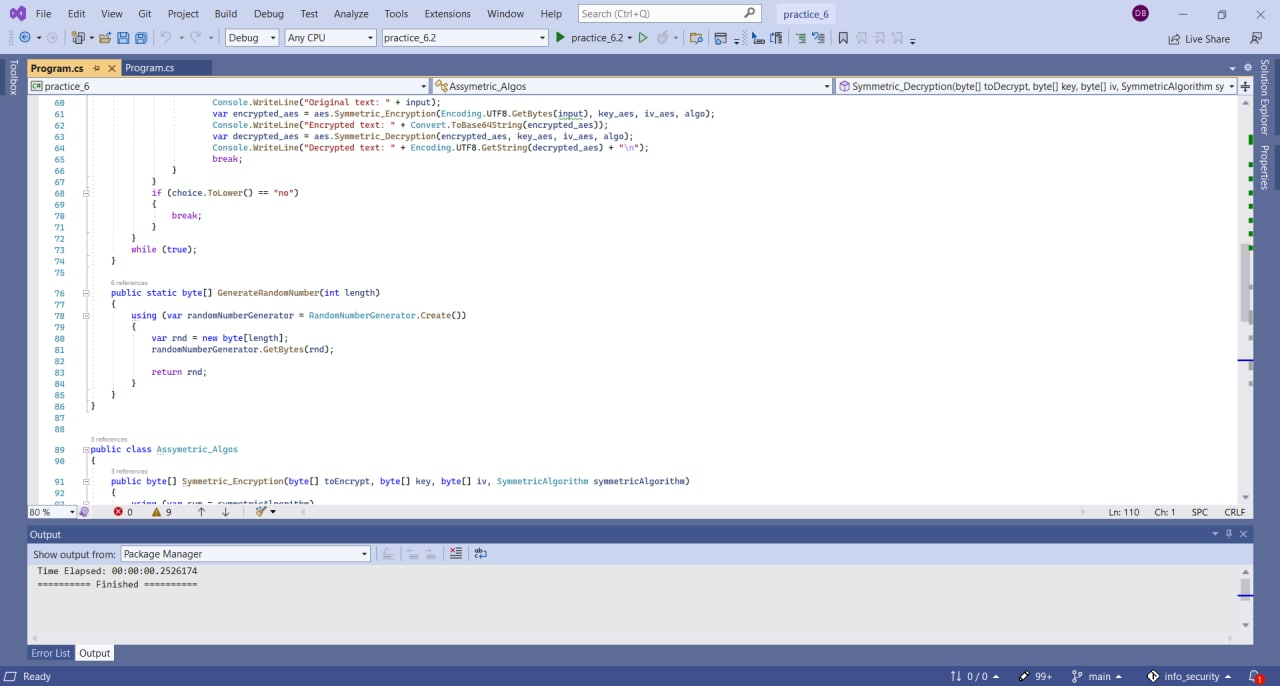
1. Напишемо інтерфейс користувача, за допомогою якого користувач зможе обрати, який саме алгоритм шифрування використати. І одразу попросимо користувача ввести текст, який потім буде шифруватись.



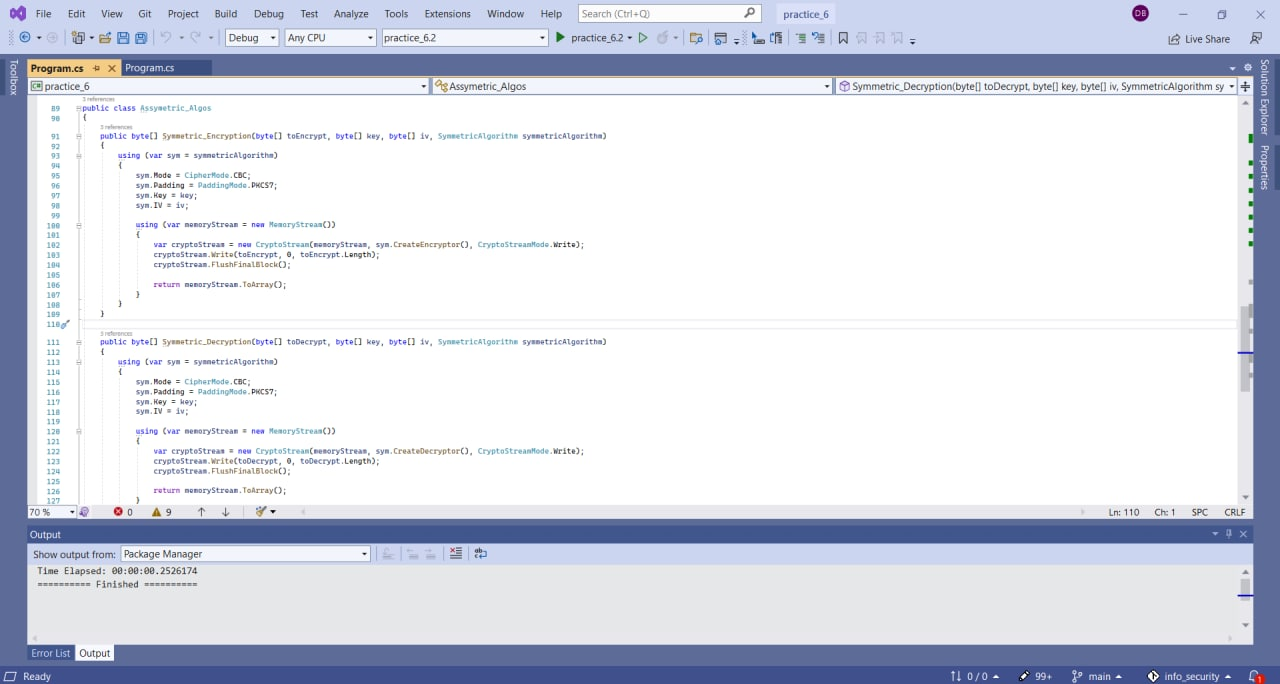
1. Далі пропишемо кейси для кожного з даних алгоритмів (дії кейсів будуть виконуватись однаково, окрім виклику алгоритму, тому розглянемо тільки один з них). Створимо об'єкт, який буде виконувати наш алгоритм, потім створимо ключ і вектор ініціалізації для нашого алгоритму відповідного розміру (8 і 8 для DES, 16 і 8 для Triple – DES, 32 і 16 для AES). Викличемо функцію шифрування та розшифрування і виведемо результати на екран.



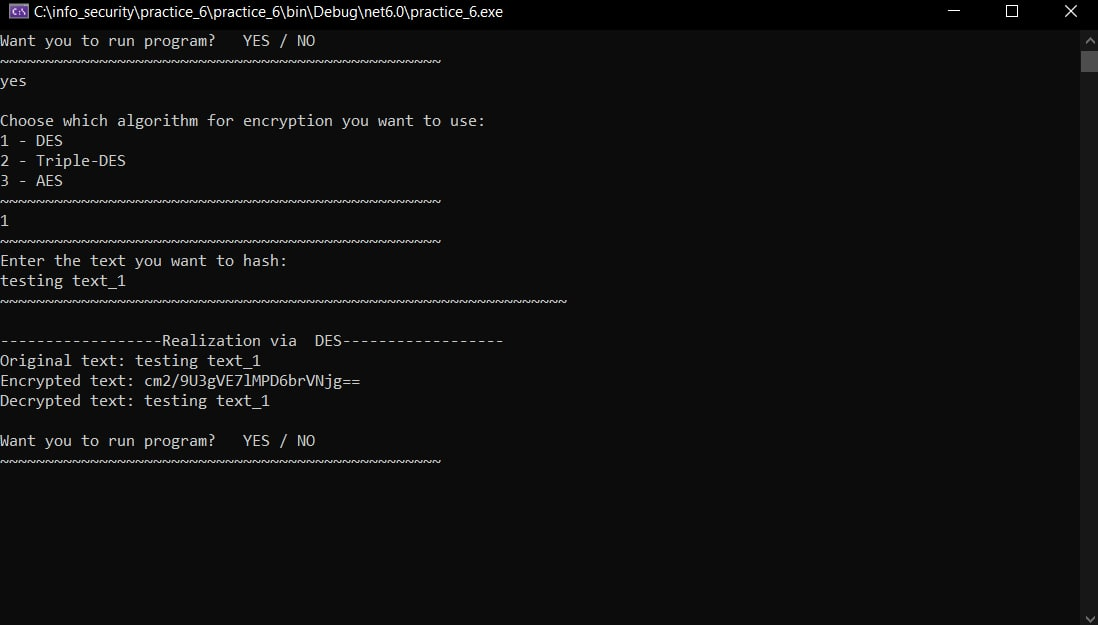
1. Пропишемо функцію генерації випадкової послідовності чисел для створення наших ключів і векторів ініціалізацій.

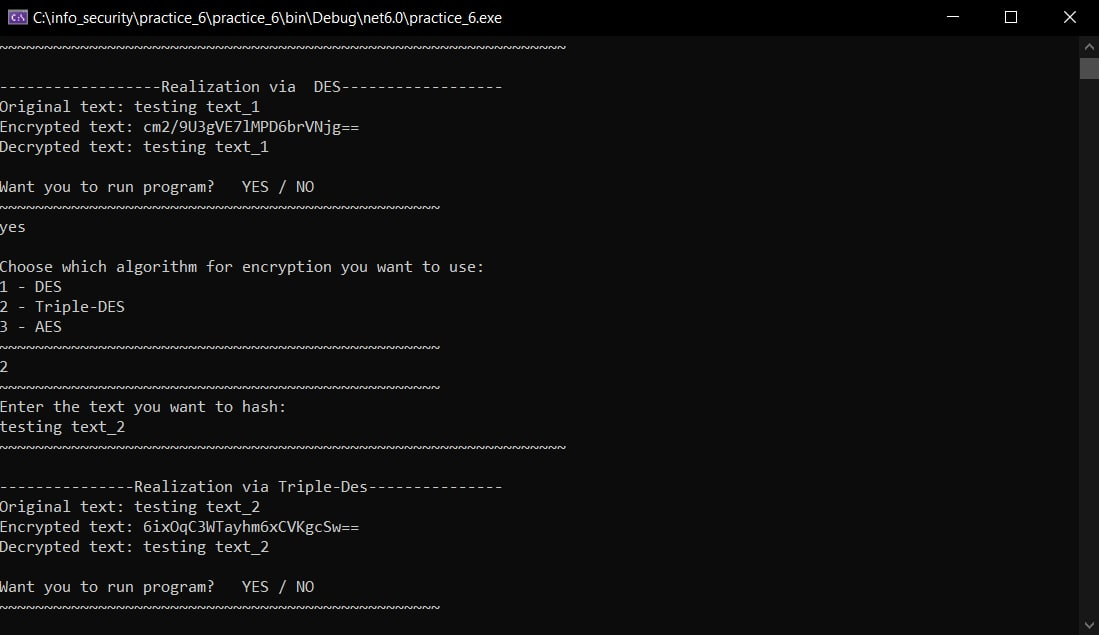


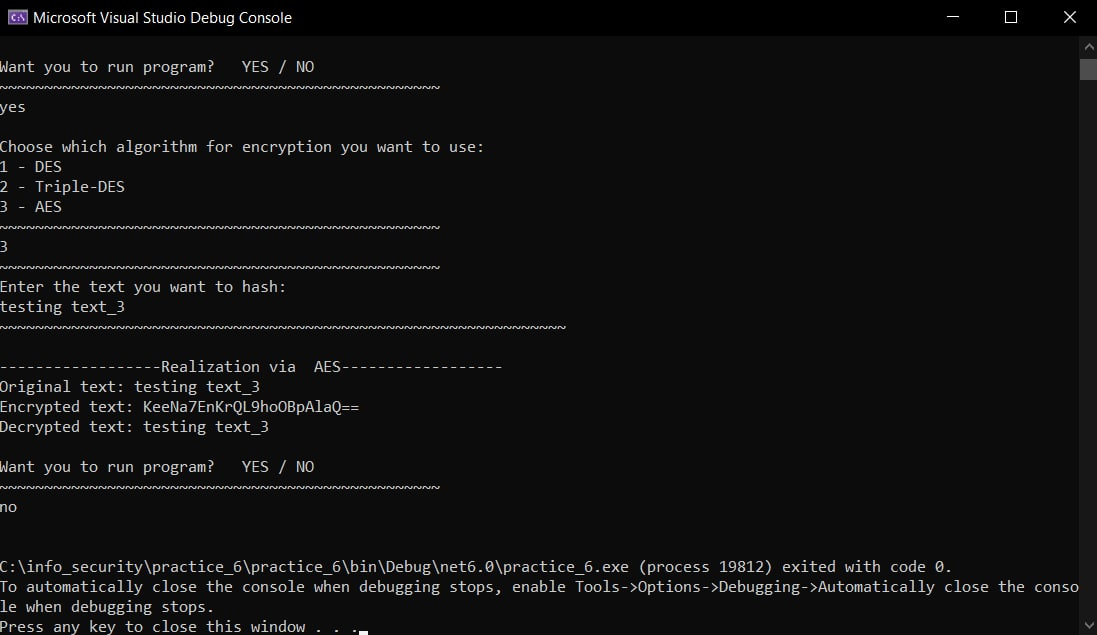
1. Створимо окремий клас для написання функцій шифрування і розшифрування. Обидві вони схожі за написанням, окрім єдиного об’єкту, який налаштовує наш потік даних із налаштуваннями на шифрування (.CreateEncryptor()) та розшифрування (.CreateDecryptor()). Спочатку запишемо властивість, яка задає режим: ми використаємо режим CBC, який використовує операцію виключного АБО для шифрування даних. Далі запишемо властивість заповнення, яка застосовується у випадку, коли довжина даних, що шифруються, менша за загальну кількість байтів, необхідних для операції (оберемо значення за замовчуванням – PKCS7). Передамо наші значення ключів і вектора ініціалізації. Далі створюємо екземпляри MemoryStream і CryptoStream для зберігання даних у пам'яті. В кінці перейдемо безпосередньо до шифрування/ дешифрування. Викличемо об'єкт з налаштуваннями для шифрування/ дешифрування, викличемо Write, який виконує операцію шифрування/ дешифрування і FlushFinalBlock, що поновлює джерело даних з поточним станом буфера, який потім очищається і наш результат передається на вихід.



Розглянемо результат виконання програми.

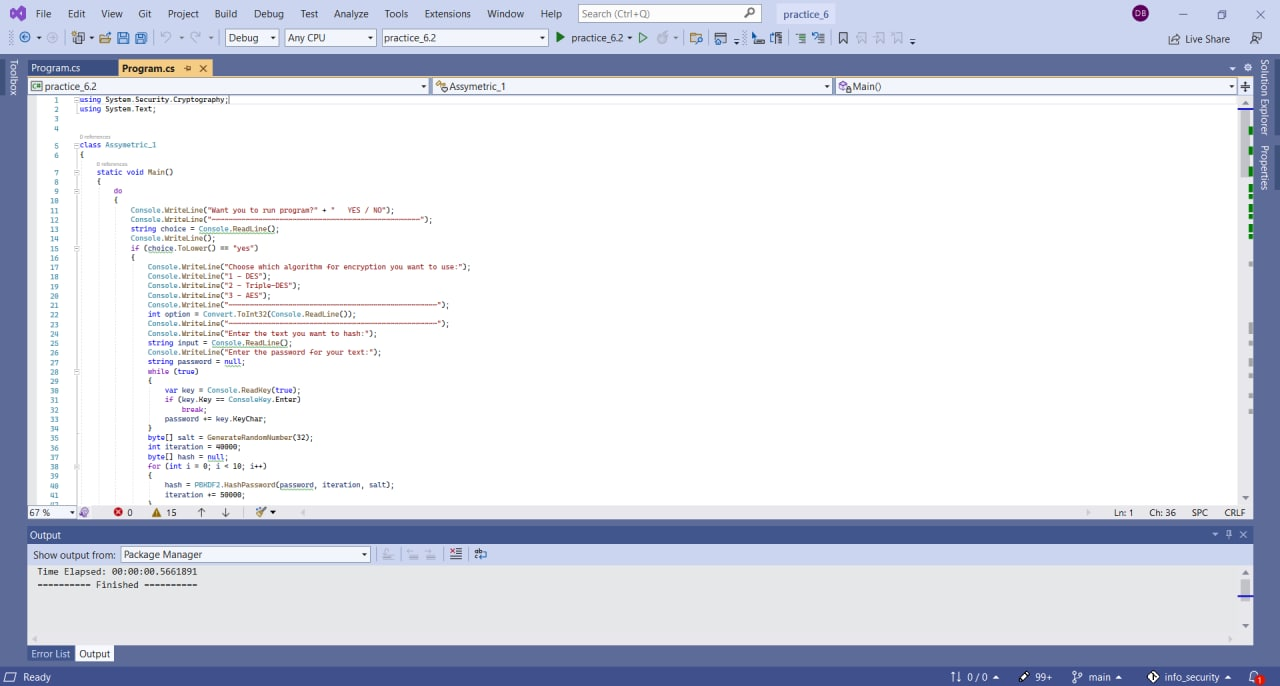


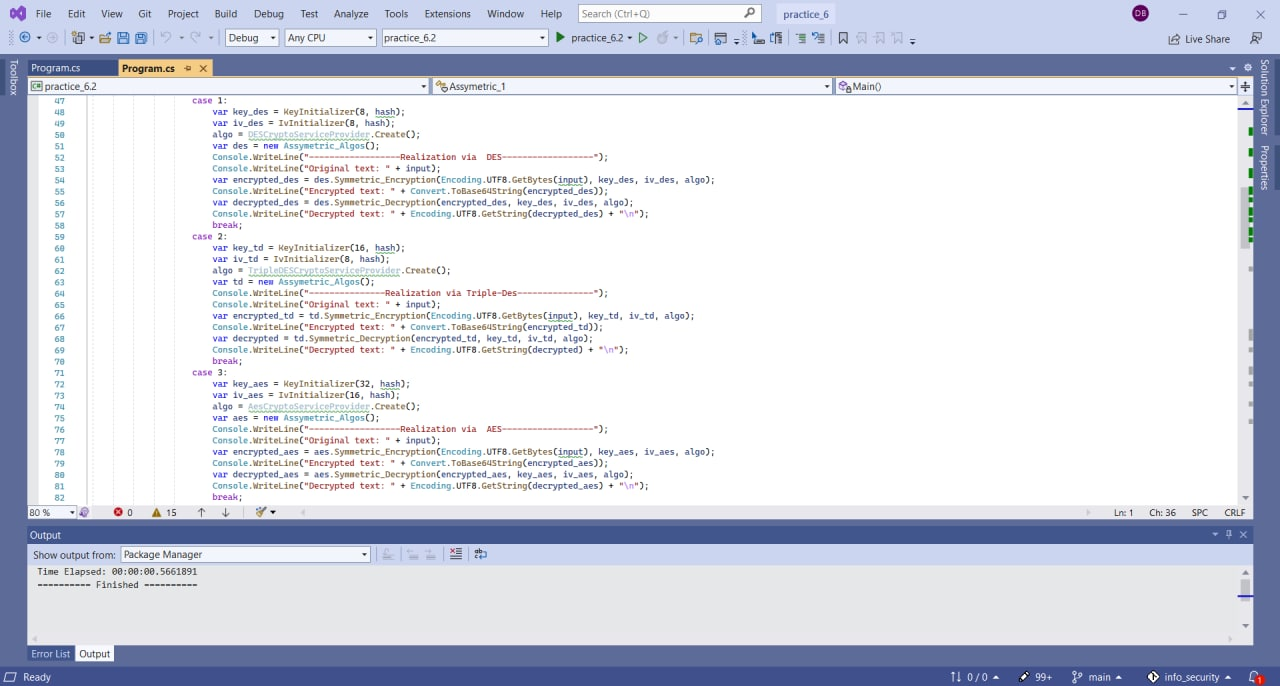


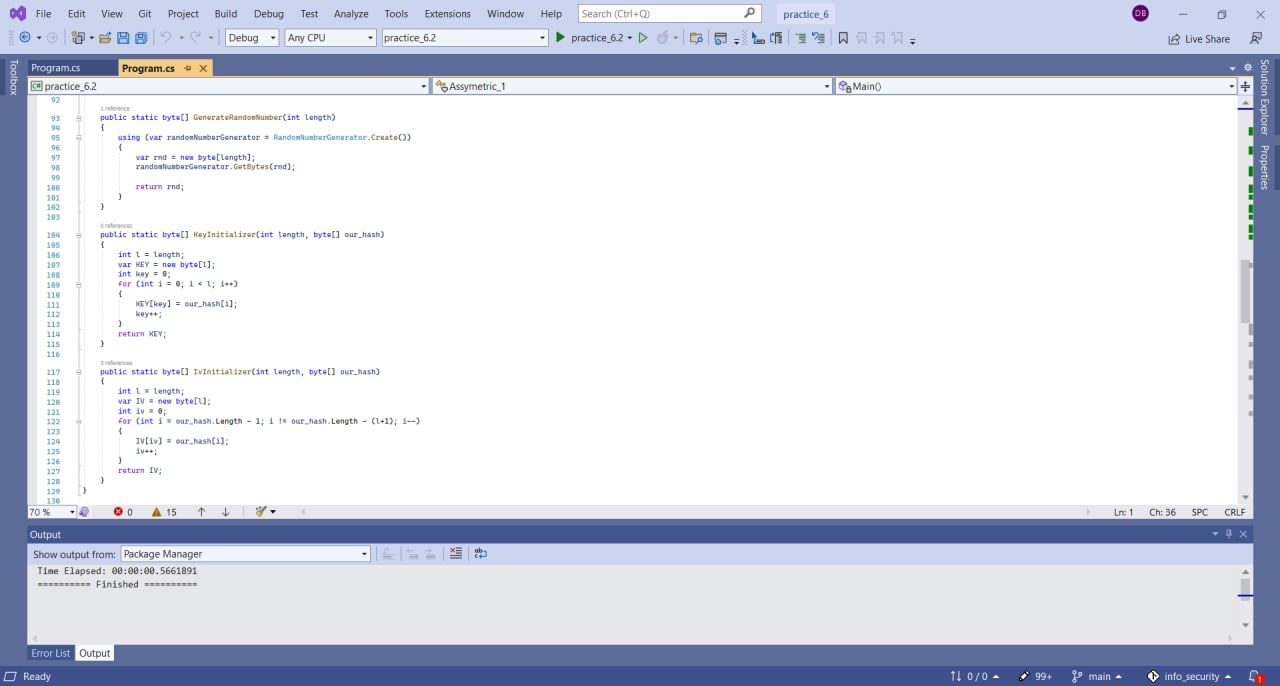


* Далі напишемо програму, що реалізує можливість задання секретного ключа та вектора ініціалізації за допомогою псевдовипадкової послідовності із використанням пароля, згенерувати сіль як випадкову послідовність байтів. У програмі використаємо клас Rfc2898DeriveBytes).

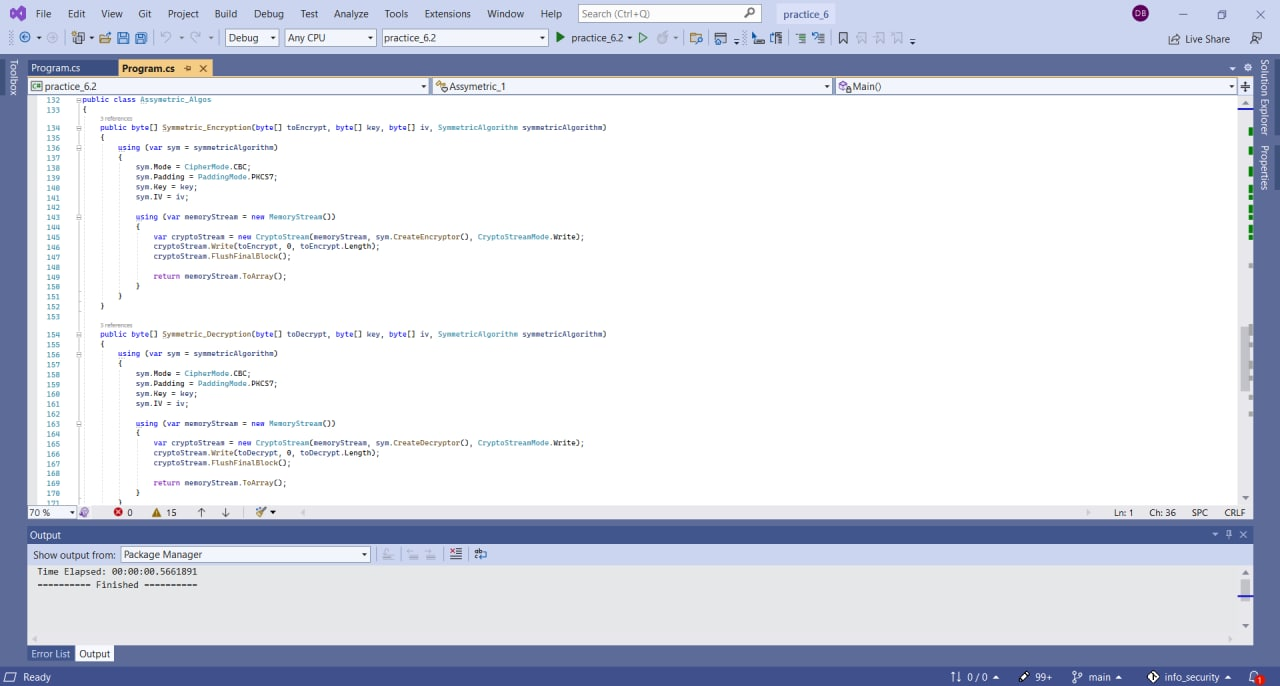
1. Напишемо інтерфейс користувача для вибору алгоритму шифрування і задання тексту та пароль для шифрування. Згенеруємо сіль і зробимо хешування заданого тексту вже відомим нам методом – додаванням додаткової ентропії, виконуючи певну ітерацію.
2. Далі пропишемо кейси, які будуть виконуватись для певного алгоритму. Всі кейси схожі за записом до кейсів з попередньої програми, крім задання ключа і вектора ініціалізації. Для їх створення ми використовуємо введений пароль: ми переводимо пароль у масив байтів і для кожного алгоритму беремо правильну кількість байтів для задання ключа (перші n байтів) і вектора ініціалізації (останні n байтів)



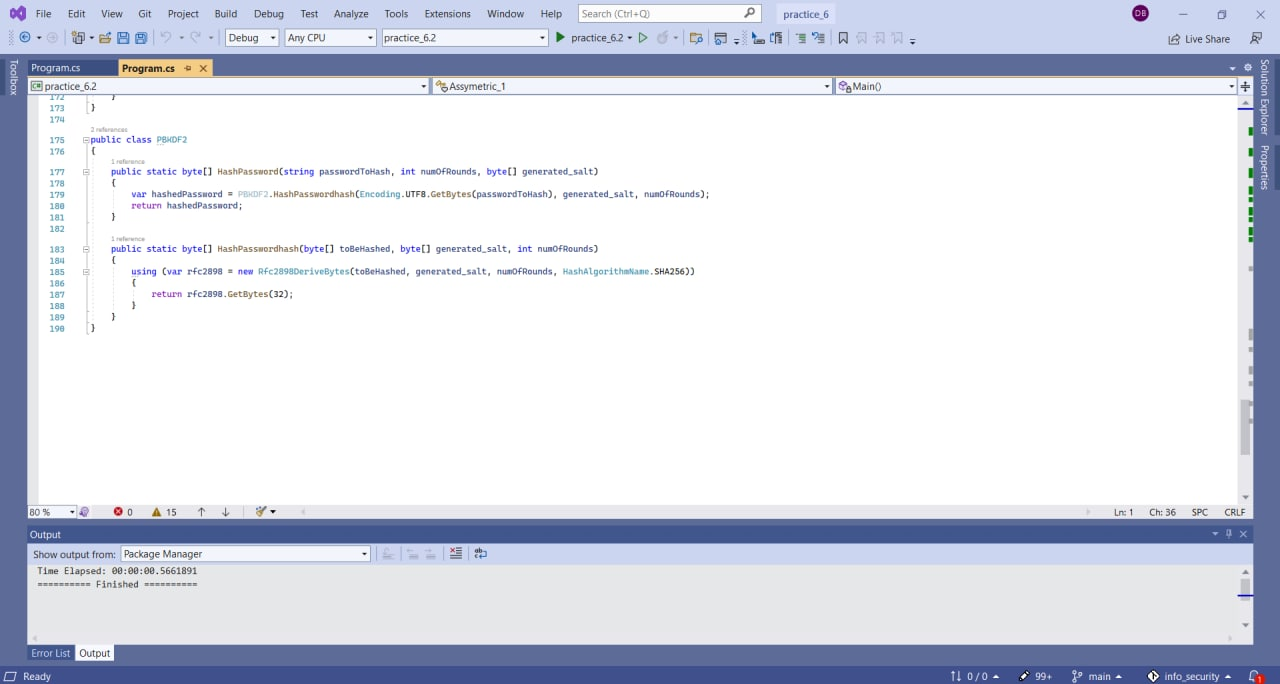




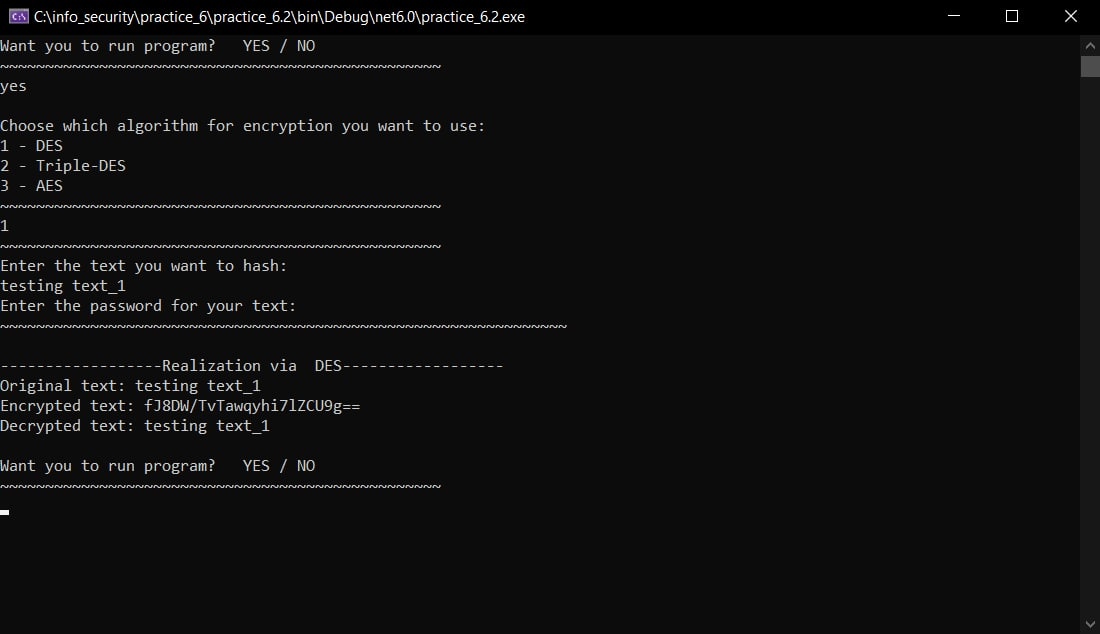
1. Далі пропишемо вже нам відомі функції шифрування/ розшифрування.

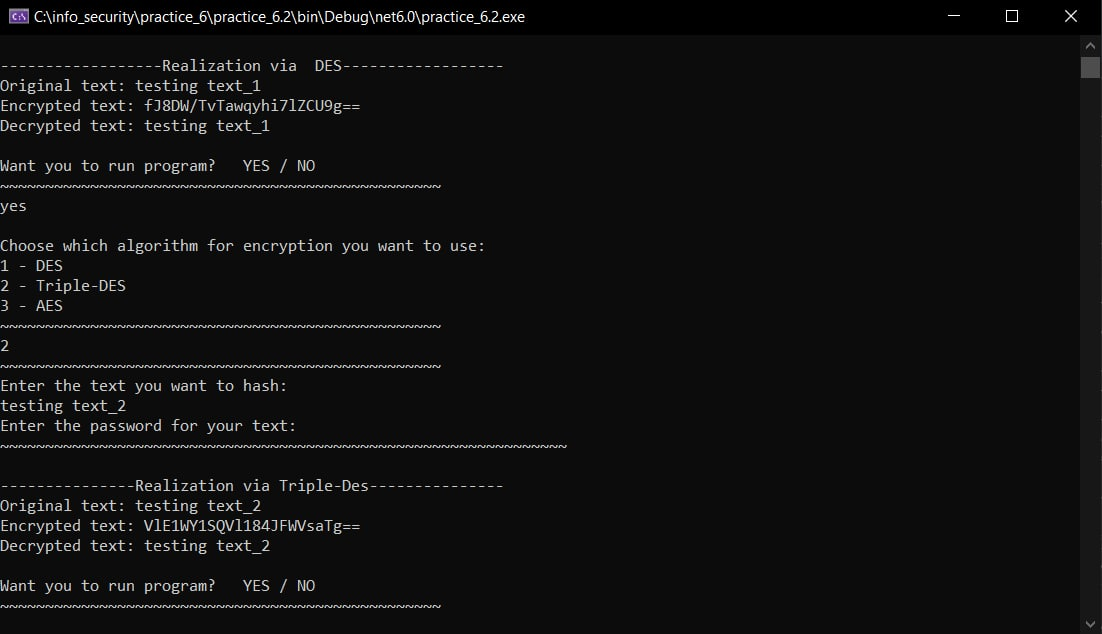


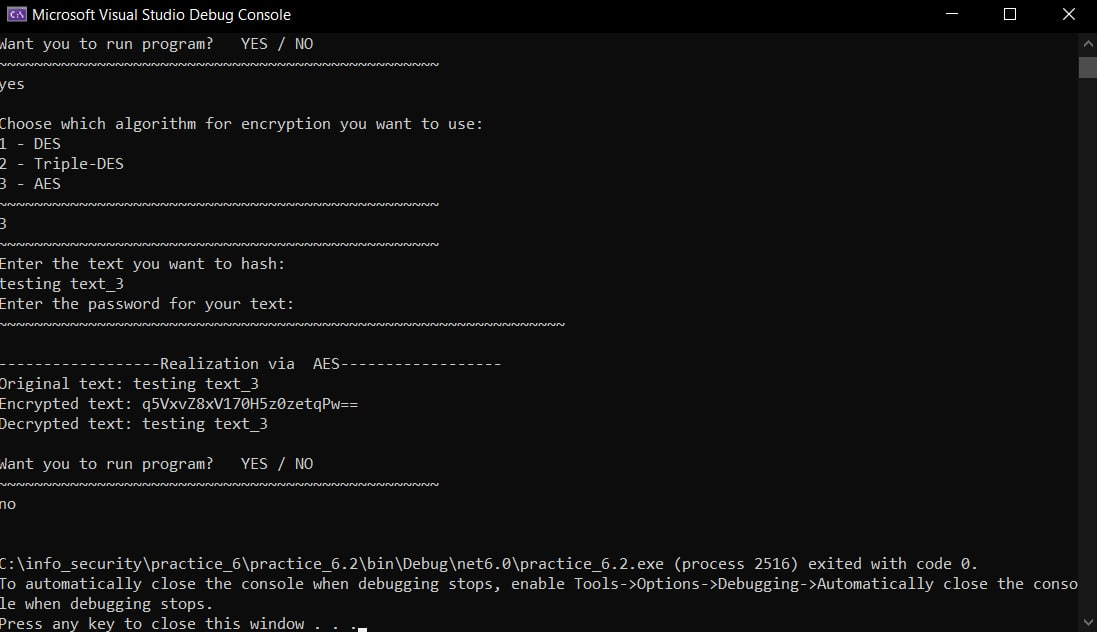
1. І в кінці програми про пишемо наш клас Rfc2898DeriveBytes, з яким ми працювали на минулій лабораторній роботі і пропишемо там функції безпосереднього хешування паролю.



Розглянемо результат виконання програми.







Висновки: у ході практичної роботи були отримані знання про симетричне шифрування, алгоритми симетричного шифрування, а саме: DES, Triple – DES, AES і навички написання цих алгоритмів та роботи з ними.