***Практичне заняття №1***

*«Генерація послідовності випадкових чисел»*

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

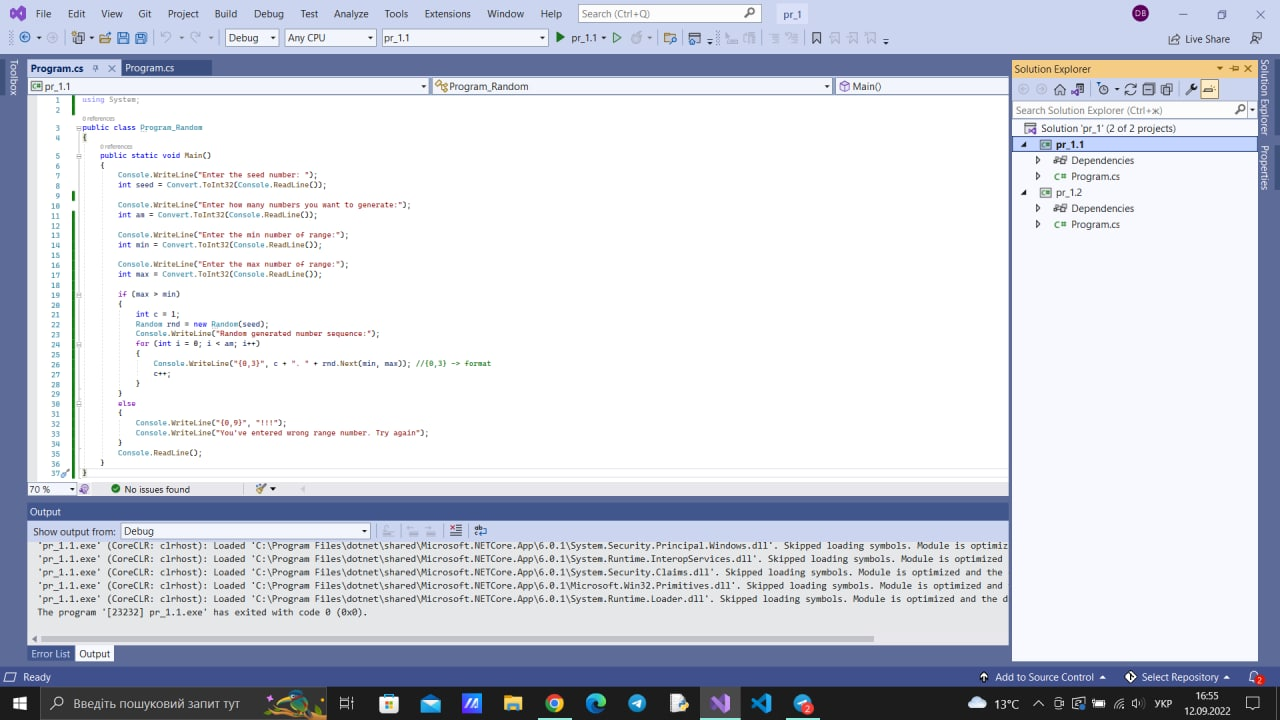
Мета: зрозуміти важливість випадкових чисел. Опанувати навички написання програм для генерації псевдовипадкових чисел і криптографічно стійкої послідовності випадкових чисел мовою С#.

Хід роботи:

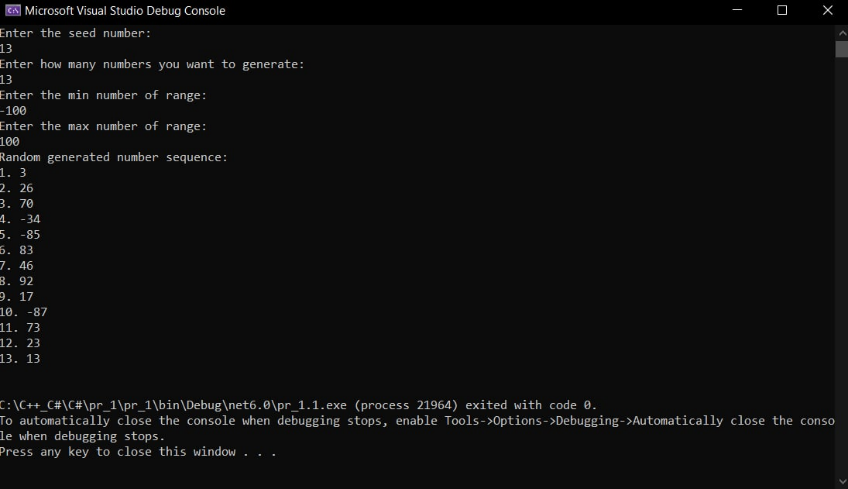
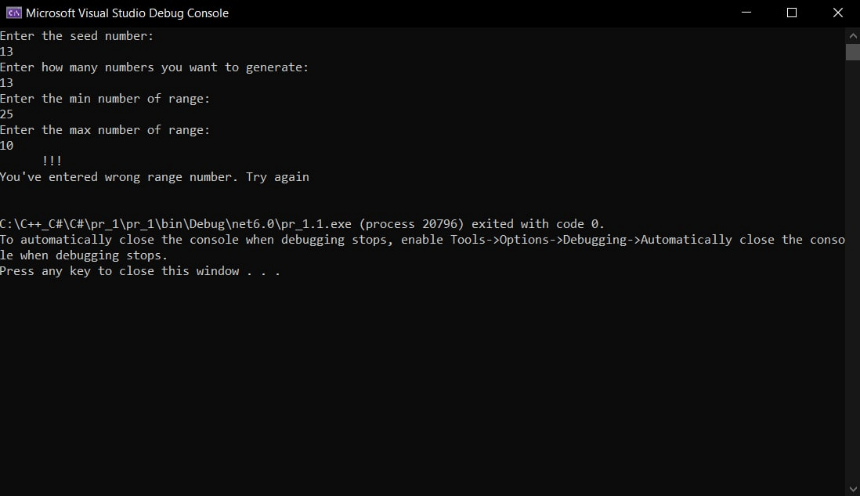
* Першим ділом створимо консольний проект у середовищі Visual Studio. Далі у проекті створимо 2 солюшн (Solution): перший для першої програми, другий – для другої.

Розпочнемо з першої програми, а саме для генерації псевдовипадкових чисел.

1. Для початку підключимо простір імен System, створимо свій клас з назвою Program\_Random і створимо свій метод, який обов’язково має називатися Main, і в ньому будемо писати свою програму, яка буде виконуватись у разі запуску проекту.
2. Для генерації випадкових чисел нам треба використати клас Random, який базується на seed значенні (вводиться користувачем). Залежно від його значення ми будемо мати різні результати програми.
3. Спочатку ініціюємо генератор певним початковим значення за допомогою оператора new. Для генерації нашої послідовності ми створюємо цикл for, який буде виконуватись n разів (n задає користувач) і виводити відповідні випадкові числа. За допомогою методу Next() класу Random ми отримаємо наші випадкові числа. Для того, щоб задати діапазон бажаних випадкових чисел, у дужках методу Next() записуємо 2 числа через кому, в межах яких будуть наші числа. В нашому випадку ці значення вводить користувач. Для правильності роботи програми робимо перевірку, щоб мінімальне число діапазону було менше максимального, використовуючи умовні твердження if та else. Якщо умова виконується, відбувається наш цикл for, якщо ні – виводимо повідомлення про неправильно введенні дані.
4. Нижче наведено скріншот з кодом, який отримали у ході написання програми і варіанти виконання програми.



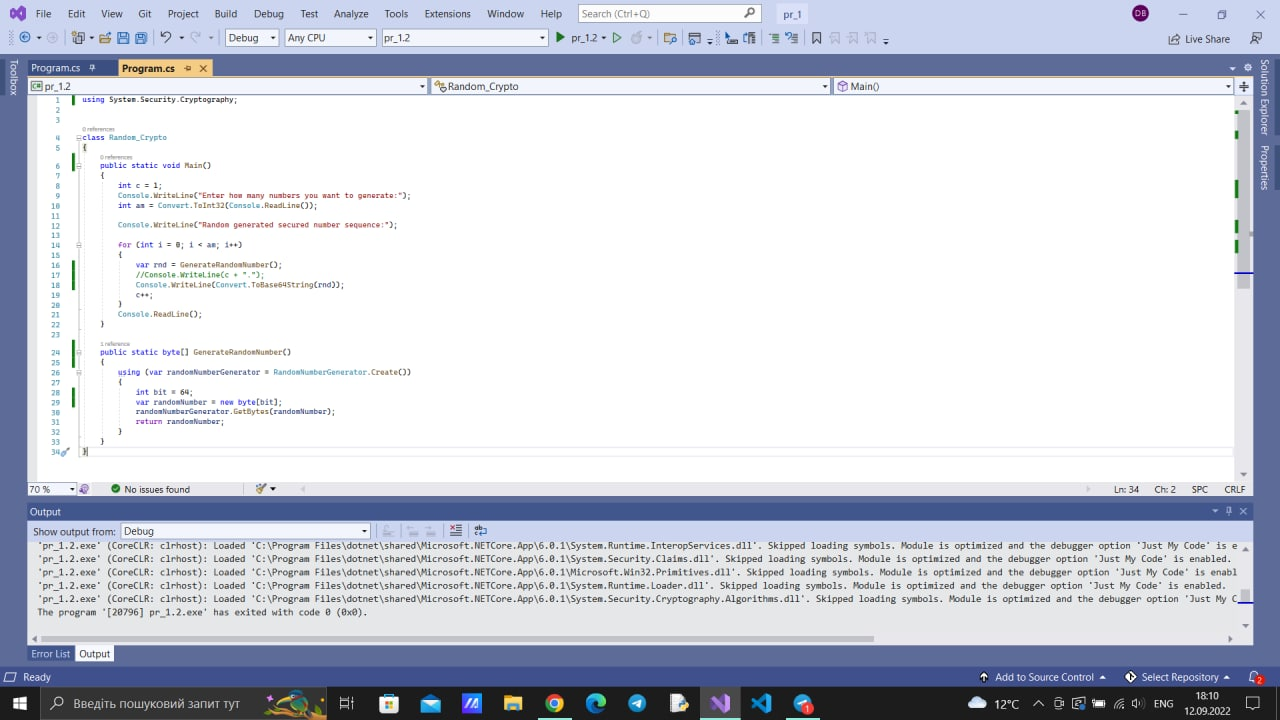
Коли умова щодо мінімального значення діапазону виконується

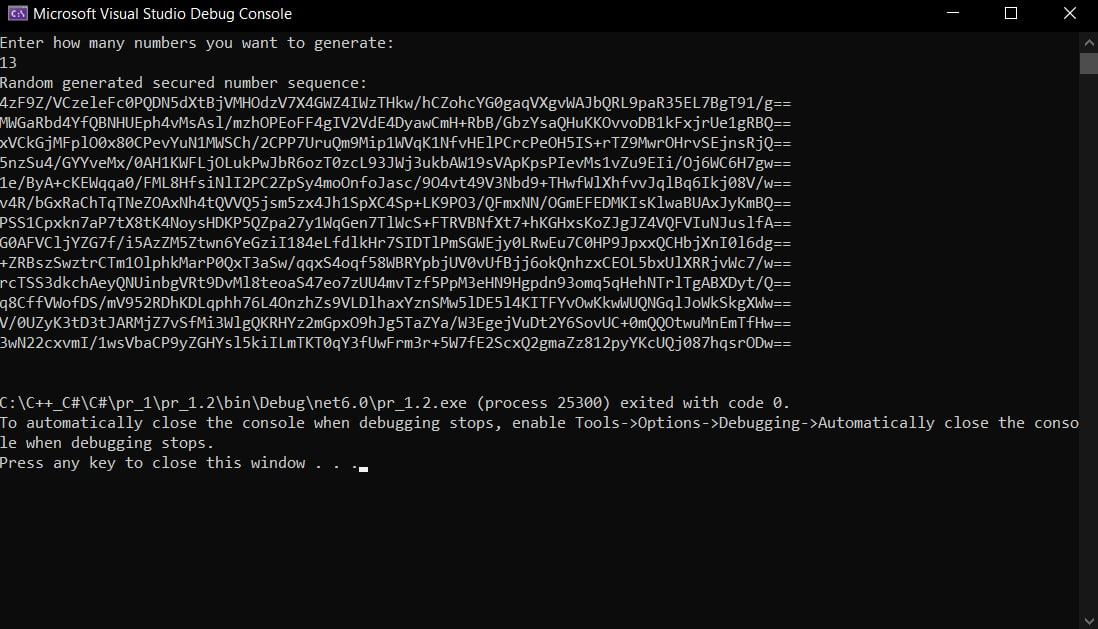


Коли ця умова не виконується (мінімальне значення діапазону більше за максимальне)

Перейдемо до написання другої програми – для генерації криптографічно стійкої послідовності випадкових чисел.

1. Для того, щоб наша послідовність була зашифрована нам треба використати метод RandomNumberGenerator.Create. Для його використання ми підключаємо простір імен System.Security.Cryptography. Потім створюємо свій клас Random\_Crypto і обов’язково метод Main.
2. Спочатку створюємо метод GenerateRandomNumber(), який міститиме в собі масив байтів. У цьому методі створюємо об’єкт методу RandomNumberGenerator.Create(). Далі створюємо сам масив байтів, числа якого будуть за довжиною заданим певним числом (в нашому випадку 64). Далі викличемо метод GetBytes і передамо в нього створений масив. Цей масив згодом заповниться випадковими даними (числами).
3. Далі просто створюємо цикл для виводу нашої генерації, який буде виконуватись n разів (n задає користувач). Також створимо об’єкт, який буде викликати наш метод створення масиву байтів і відповідного значення випадкового числа.
4. При запуску програми отримаємо не дуже читабельний результат, та для того, щоб конвертувати масив байтів у рядок, ми можемо викликати метод Convert.ToBase64String. Таким чином отримаємо читабельний результат програми.
5. Нижче наведено скріншот з кодом, який отримали у ході написання програми і варіант виконання програми..

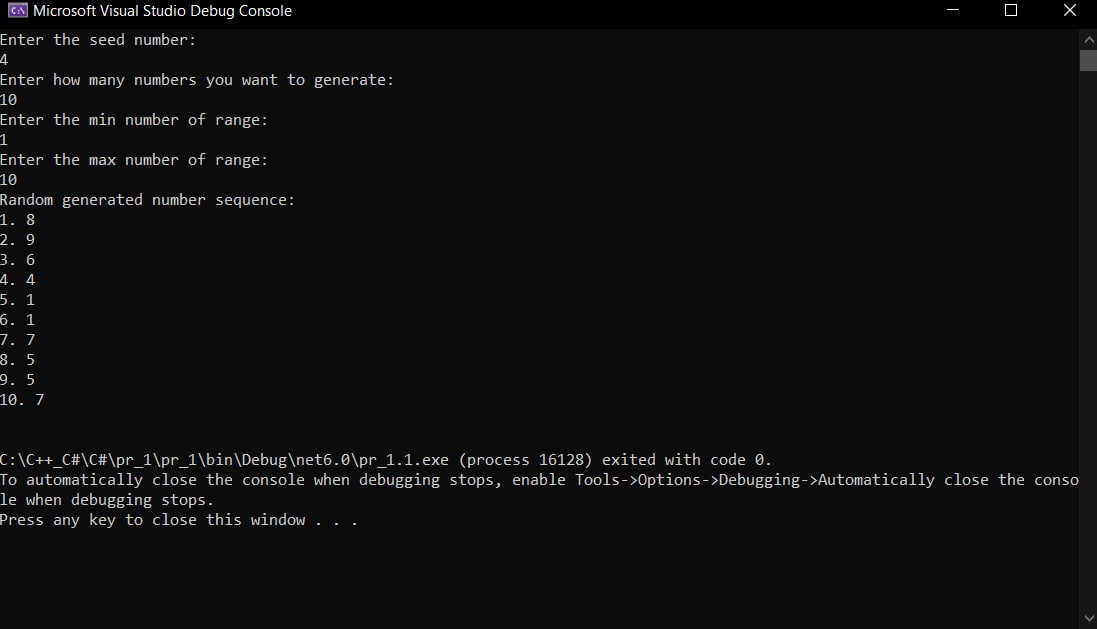




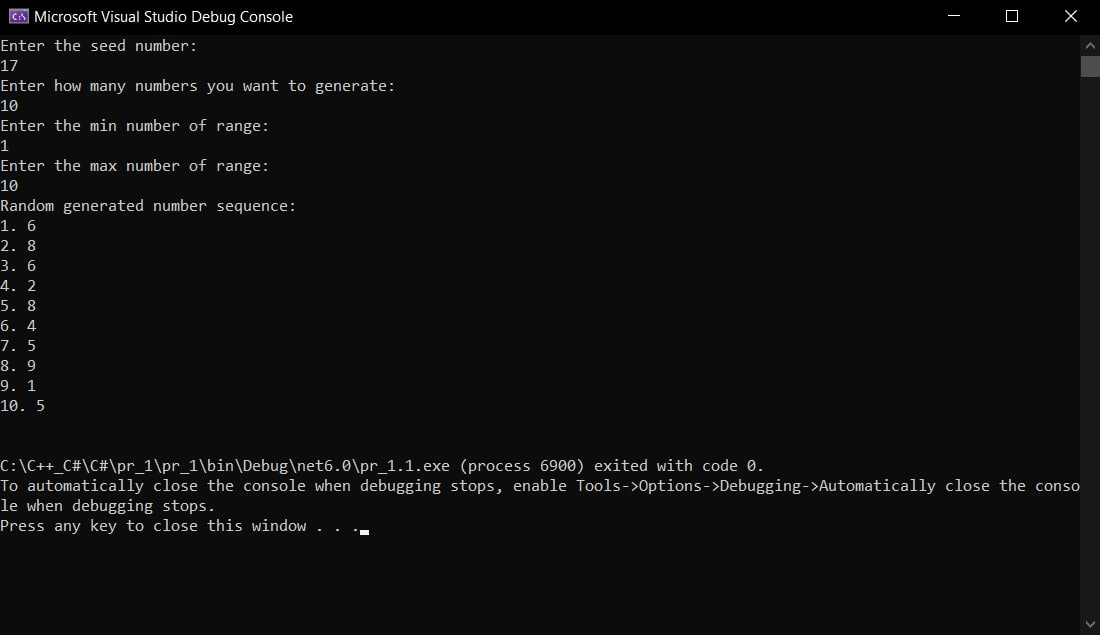
* Тепер порівняємо результати виконання з різними введеними даними обох написаних програм.

Розпочнемо з першої програми.

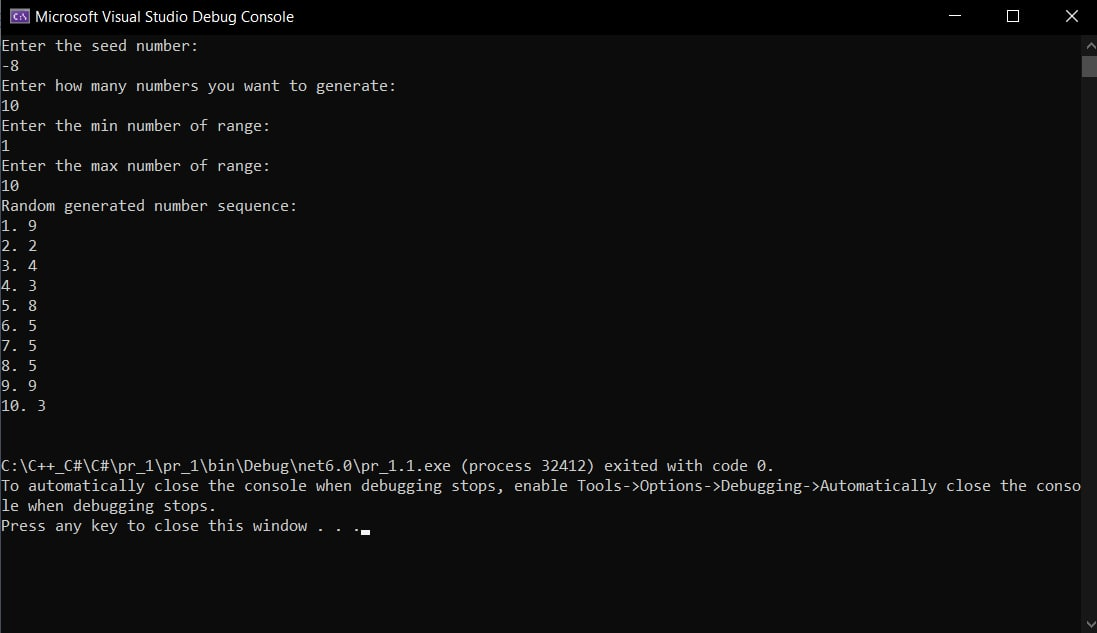
Нехай перше seed значення буде 4, маємо такі результати:



Візьмемо інше значення – 17:

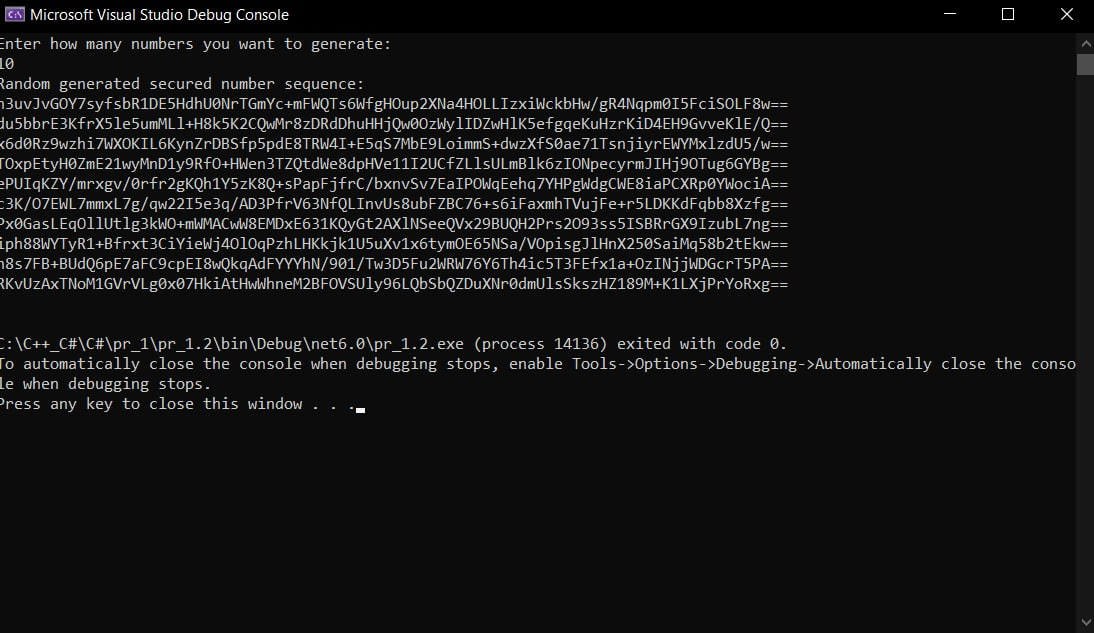
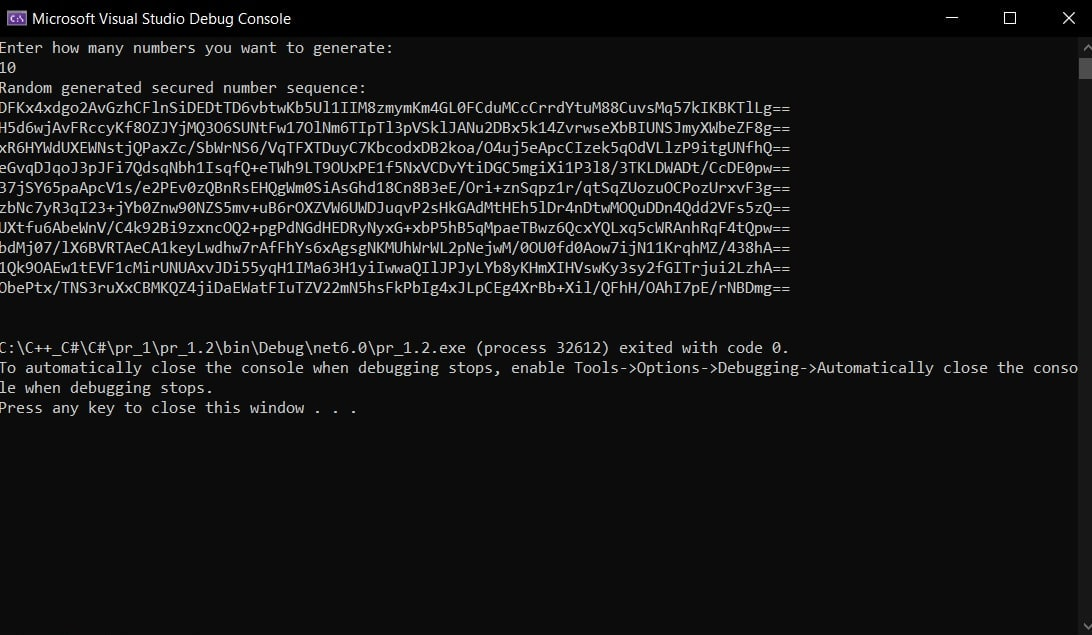


Візьмемо ще одне значення – -8:



Розглянемо другу програму.

Числа лишимо ті ж самі, але подивимось різницю результатів виведення:

1. 
2. 

Висновки: у ході практичного заняття були отримані навички писання мовою С#, а саме: вміння написати програми для генерації послідовності випадкових чисел і криптографічно стійкої послідовності випадкових чисел. У ході результатів програм було дійдено таких висновків – у першій програмі головну роль грає seed значення: змінюючи його, змінюється результат виконання; у другій програмі за рахунок шифрування даних кожен раз виводиться різний результат. Якщо дивитись з точки зору захисту і безпеки даних, то доцільніше використовувати криптографічно стійку послідовність, адже вона дає більший захист від усяких загроз.