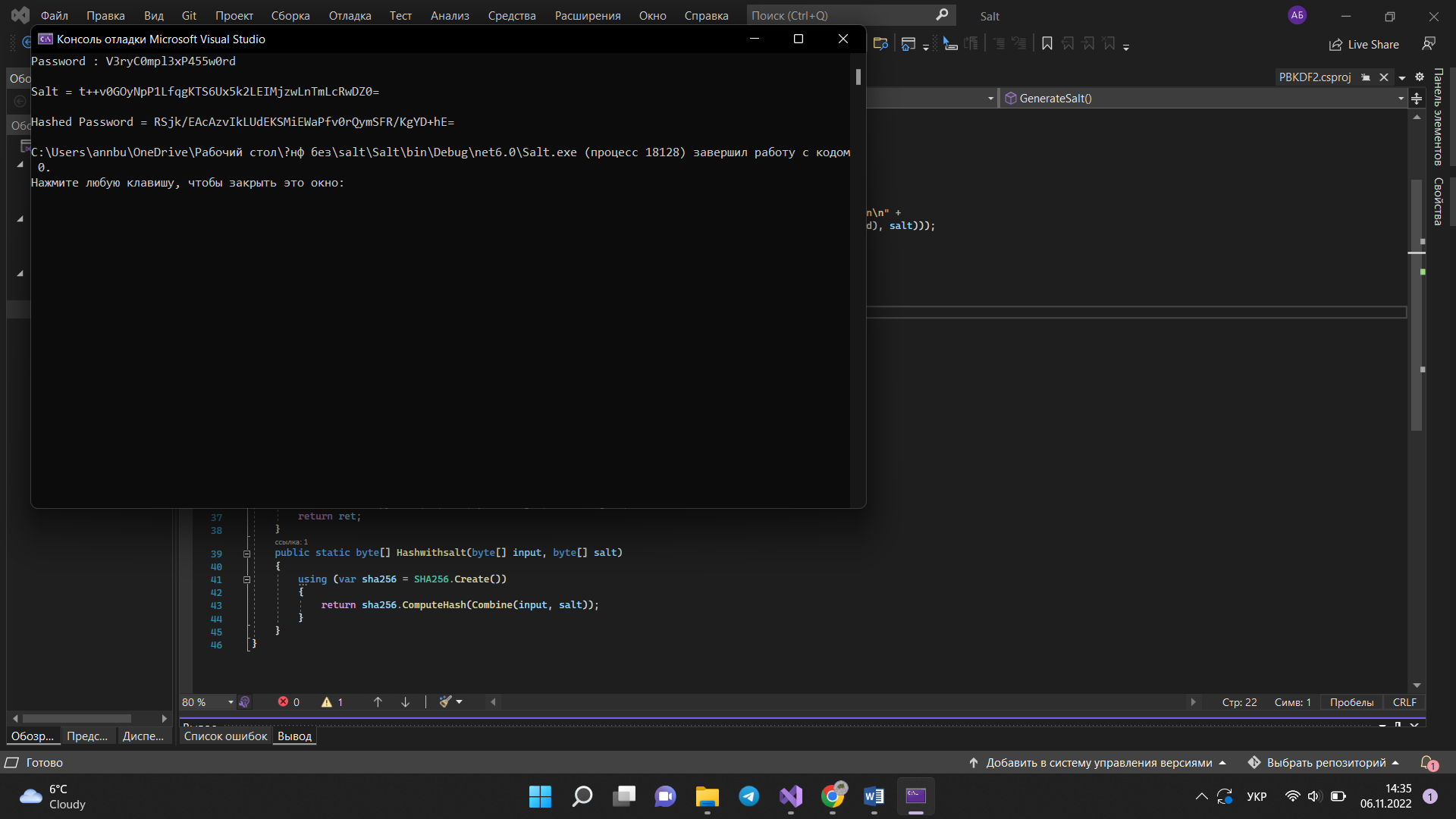
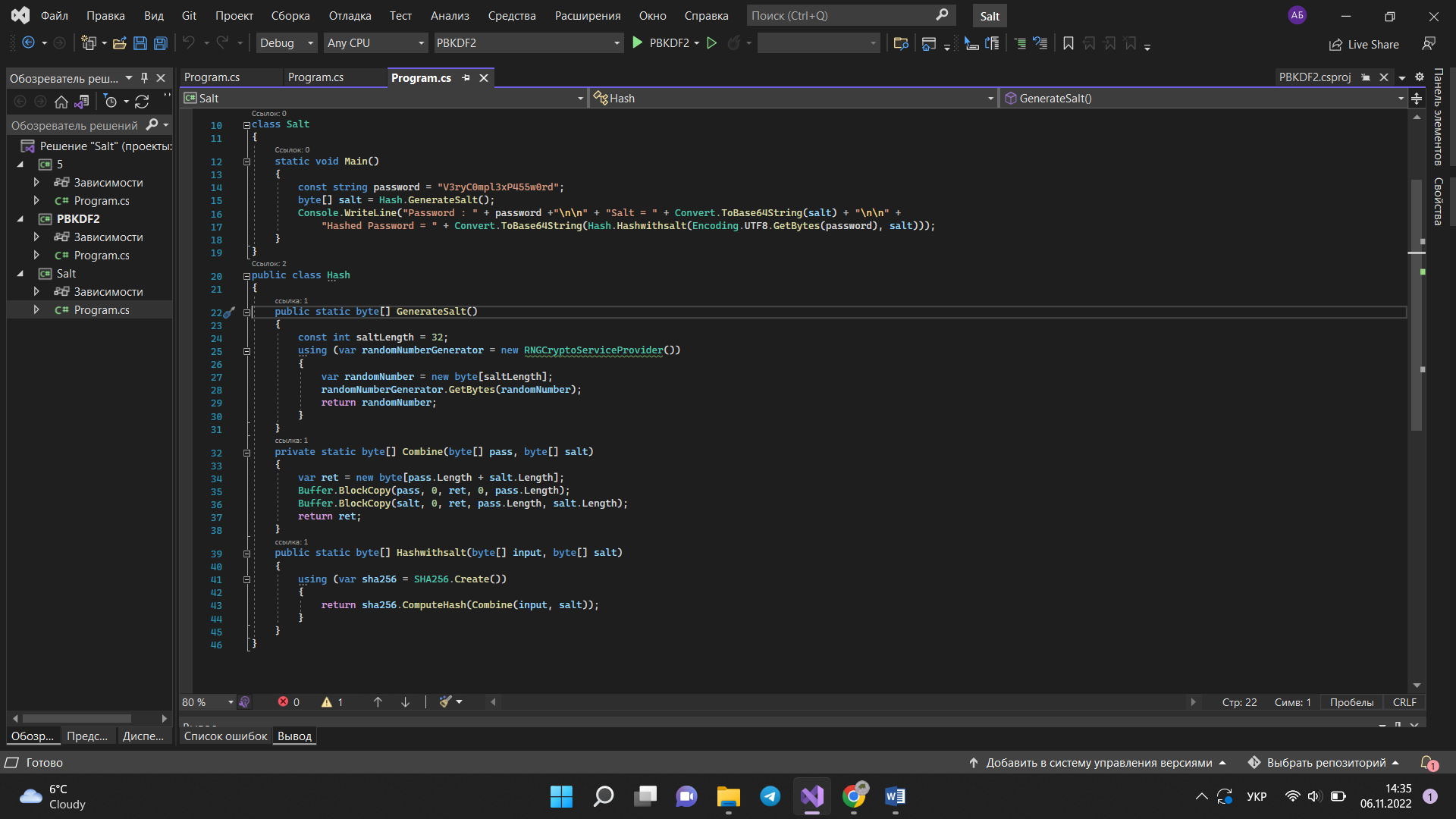
***Практичне заняття №5***

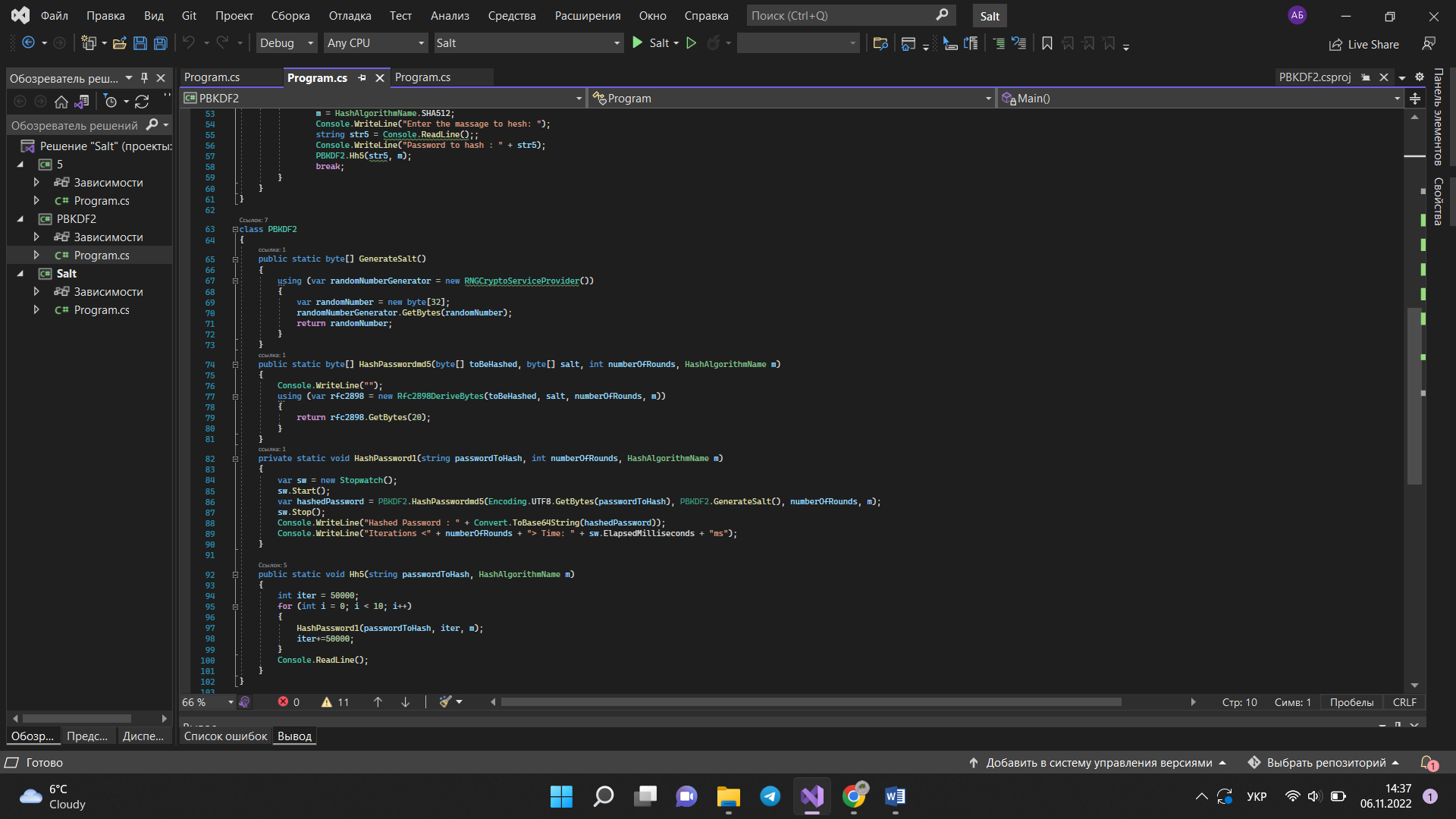
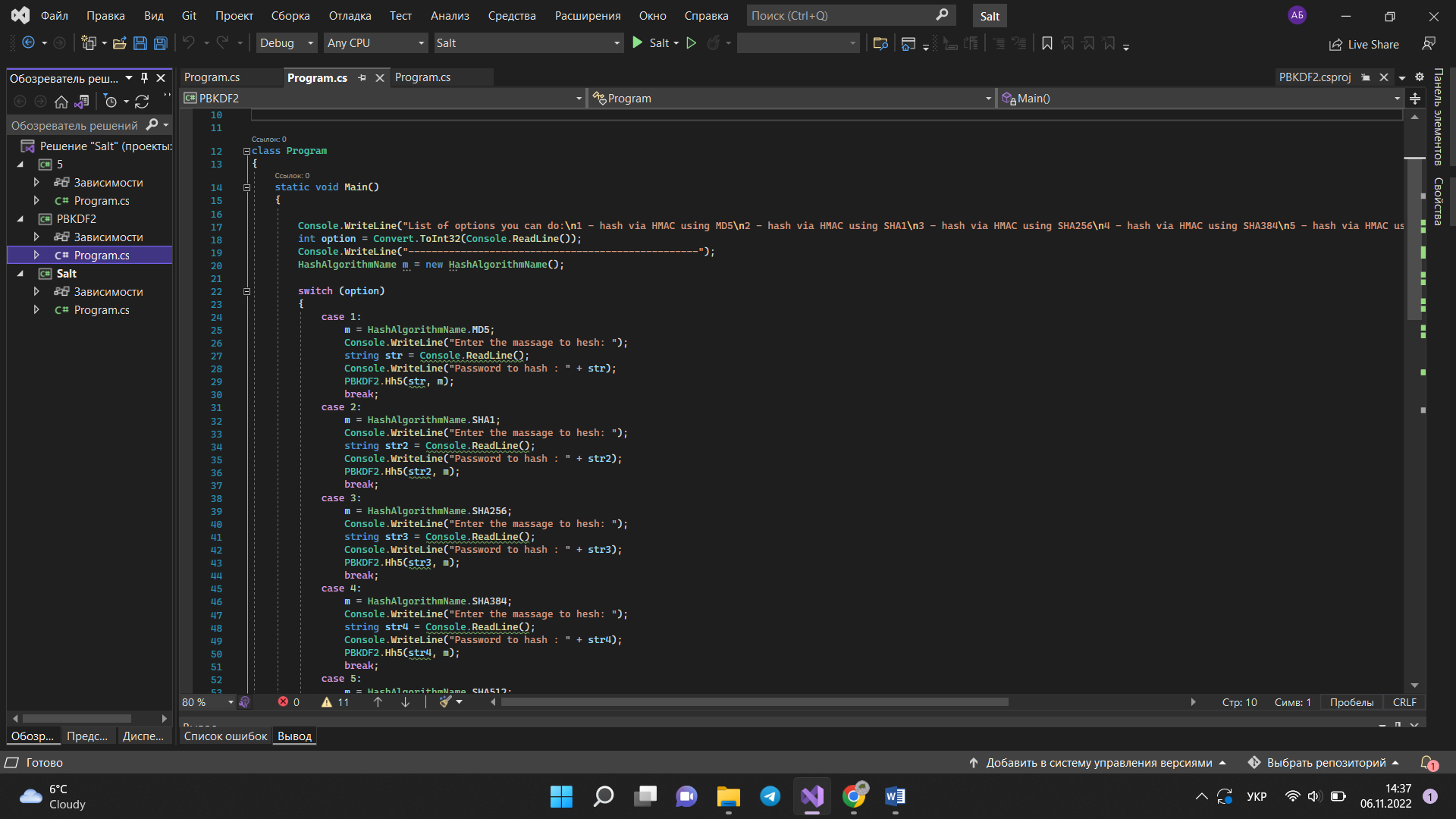
*«Безпечне зберігання паролів»*

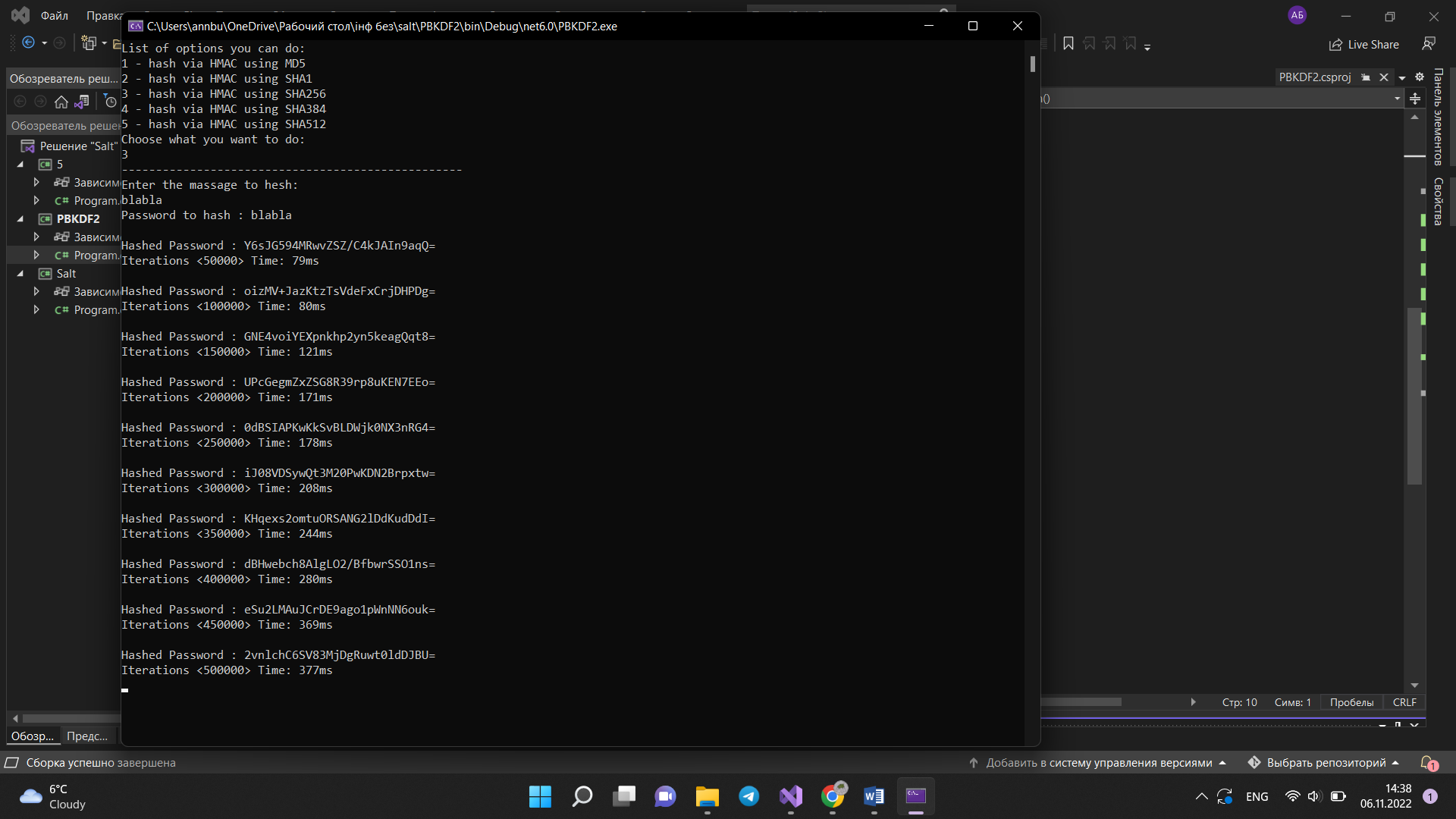
Виконала студентка групи МІТ-21 Булгакова Анна

3. Розробити клас SaltedHash, що реалізує хешування паролів із додаванням додаткової ентропії. Продемонструвати роботу класу, обчислюючи хеш для заданого пароля та "солі".

Виконавши третє завдання ми практично застосували солене хешування, при якому збільшуємо ентропію вихідного пароля шляхом генерації випадкового числа, яке додаємо до пароля перед хешуванням. Та можемо зробити висновок, що надійніше використовувати солене хешування чим пряме, бо процес соління значно ускладнює використання таблиць грубого перебору та райдужних таблиць. Використовувати даний спосіб це хороший варіант, який на сьогоднішній день досить часто використовують, проте є кращий, який ми використовуємо в наступному завданні.

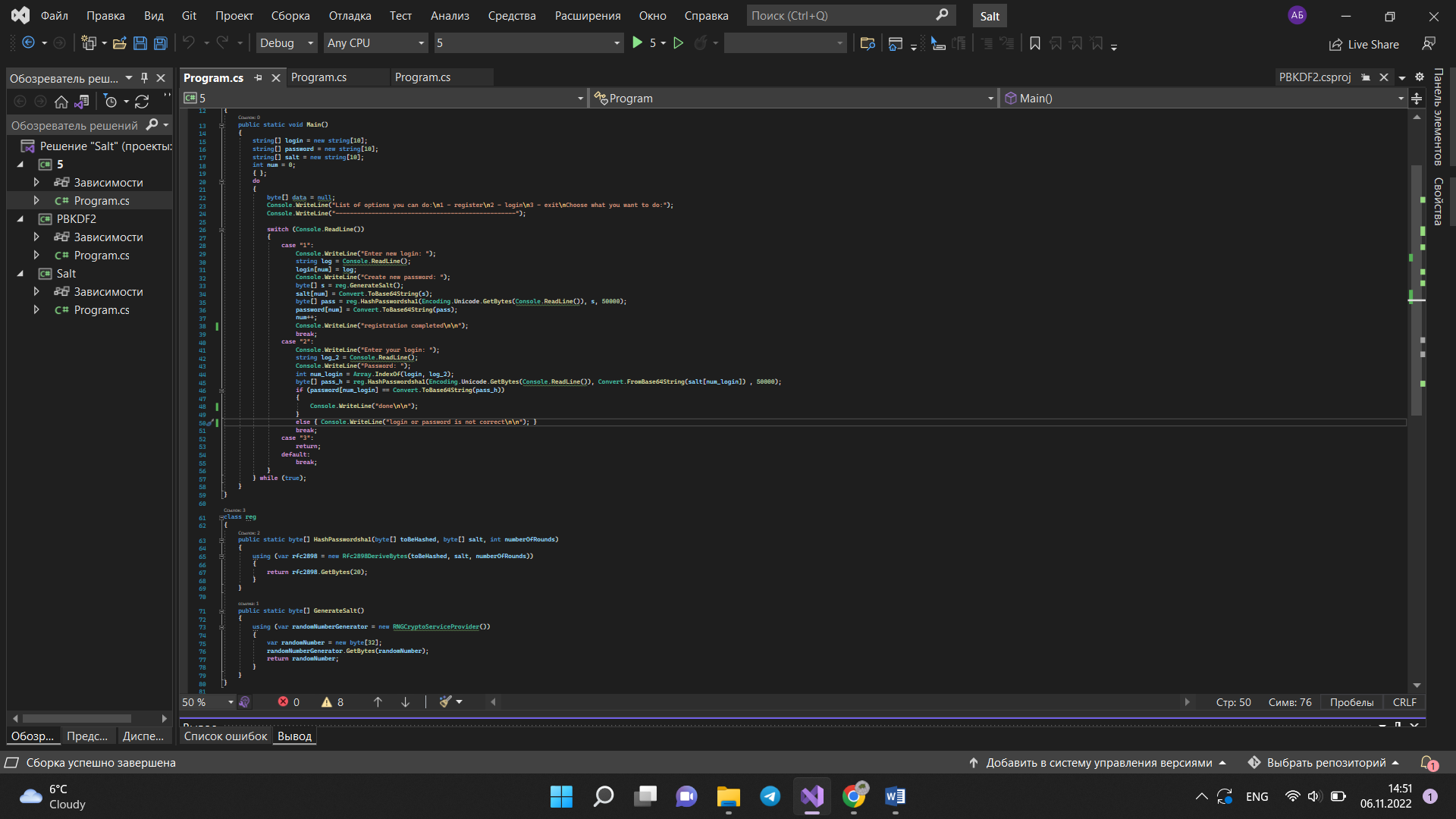
4. Розробити клас PBKDF2, що має наступну функціональність: генерує "сіль", задає алгоритм хешування (MD5, SHA1, SHA256, SHA384, SHA512) та обчислює хеш для заданого числа ітерацій. Створити програму, що обчислює час, витрачений на обчислення хешу для різного числа ітерацій (10 значень із кроком 50'000; перше значення = номер варіанта \* 10'000 ). Побудувати графік залежності витраченого часу від числа ітерацій.

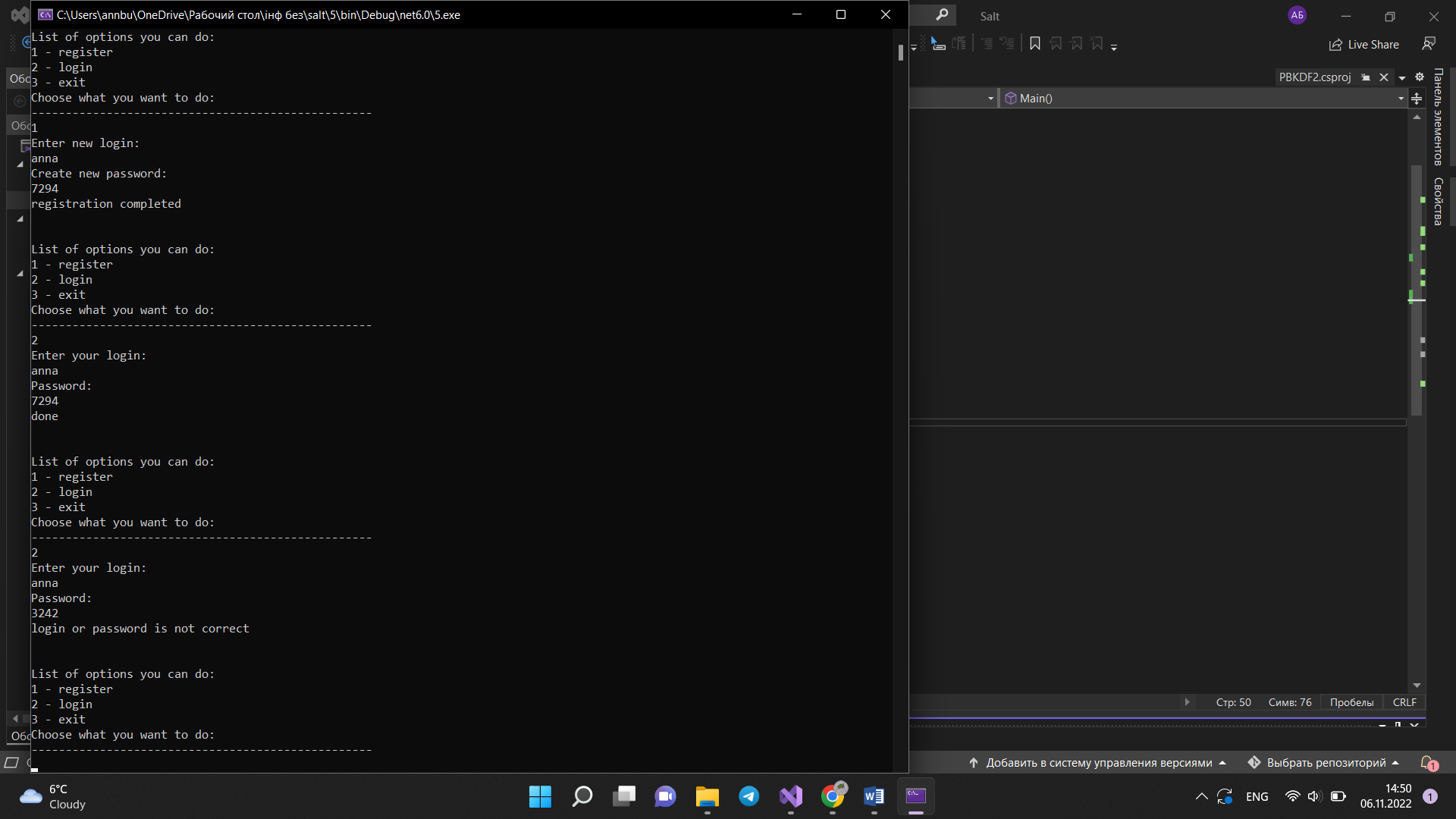




Виконавши дане завдання ми розробити клас PBKDF2, що генерує "сіль", задає алгоритм хешування та обчислює хеш для заданого числа ітерацій та створили програму, що обчислює час, витрачений на обчислення хешу для різного числа ітерацій. Процес PBKDF2 схожий на соленого хешування, але він вводить у функцію додатковий вхід, який є підрахунком кількості ітерацій. Мета цього підрахунку полягає в тому, щоб алгоритмічно уповільнити процес хешування, щоб посилити атаки грубої сили. Замість того, щоб сучасні графічні процесори намагалися обробляти мільярди хешів за секунду, функція виведення ключів на основі пароля може зменшити кількість спроб до п’яти або десяти хешів за секунду. Це робить процес злому паролів більш надійним і безпечним.

Але ми повинні пам’ятати про число, яке ви використовуєте для підрахунку ітерацій, бо чим воно більше тим більша затримка виконання хешування і про обов’язково потрібно пам’ятати, бо затримка неминуча. Це гарно видно на графіку.

5. Написати програму, що реалізує хешування введеного пароля під час реєстрації користувача та зберігає логін, пароль та "сіль" у пам'яті. Реалізувати можливість автентифікації за логіном і паролем. Число ітерацій = номер варіанта \* 10'000. 



В останньому завданні ми зробили програму, яка реалізує хешування введеного пароля за допомогою PBKDF2 під час реєстрації користувача та зберігає логін, пароль та "сіль" у пам'яті.

**Висновок:** На даній практичній роботі ми отримали знання про сіль для хешування, хешування з додатковою ентропією, використовуючи алгоритми хешування вивчені на попередній практичній роботі.