Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 3**

з дисципліни «Системне програмування» на тему

«Дослідження структури програм формату EXE»

Виконав: Перевірив:

Студент 2 курсу ФІОТ-у доц. Павлов В.Г.

групи ІМ-33

Козарезов Кирил Олександрович

номер у списку групи - 10

Київ 2025

**Мета роботи:**

Ознайомлення із специфікацією **COFF** (Common Object File Format). Вивчення прийомів дослідження структури файлів **PE**-формату

**Порядок виконання роботи:**

1. Вивчити структуру програм формату EXE [1].

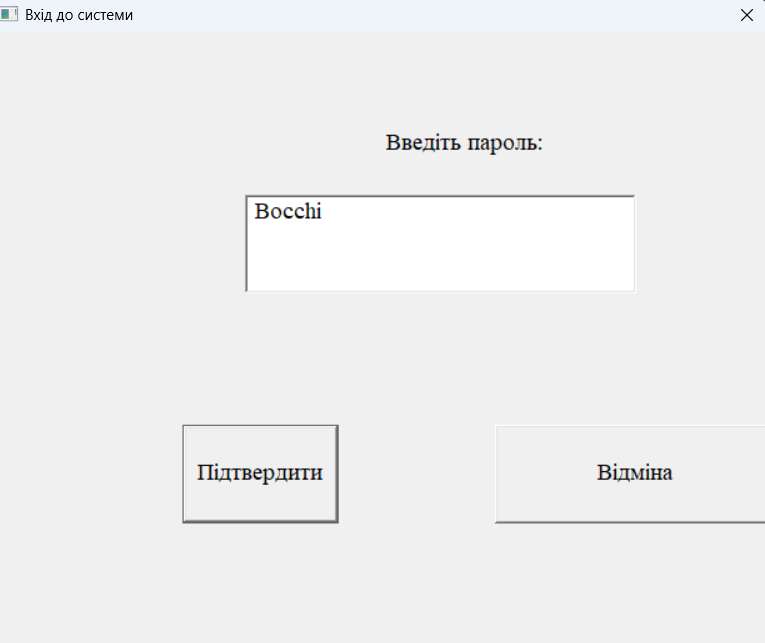
2. Розробити програму на мові Асемблер, за допомогою якої у віконному інтерфейсі виводяться персональні дані студента – ПІБ, дата народження, номер залікової книжки тощо (див. лаб. роботу 1), але лише при вірно введеному паролі

3. Виконати компіляцію розробленого файлу у формат **EXE**.

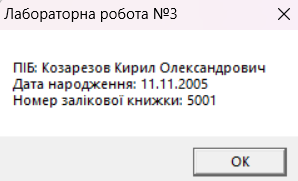
4. Перевірити роботу програми шляхом введення як правильного, так і невірного паролів:

Правильний пароль: **Bocchi**

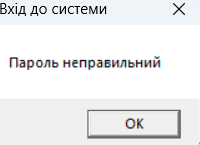
Вікно введення паролю:



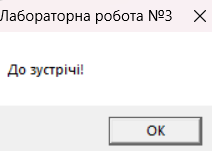
Результат введення правильного паролю:



Результат введення неправильного паролю:



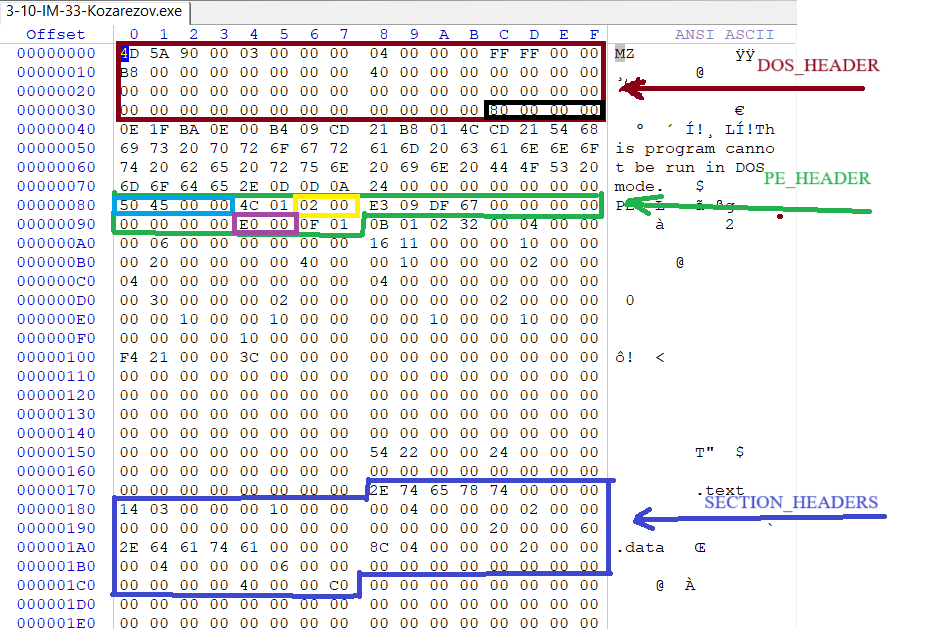
Вікно, що відкривається при натисканні кнопки «Відміна»:



5. Отриманий виконавчий файл дослідити за допомогою програми **HEX**-редактора **HIEW32** або **WinHex**

6. На скріншоте перших 25 рядків вмісту файлу обвести кольоровим олівцем або фломастером області MS-DOS заголовка (**DOS\_HEADER**), PE заголовка (**PE\_HEADER**) і таблиці секцій (**SECTION\_HEADERS**). Скріншот привести в звіті по лабораторній роботі.

Дослідження файлу в **HEX**-редакторі **HIEW32:**



**Пояснення:**

Аналіз файлу розпочався з ідентифікації **DOS-заголовка**:

* Він завжди розташований на початку файлу.
* Має фіксований розмір 64 байти (на скріншоті позначено темно-червоним).

Наступним кроком був пошук **PE-заголовка**:

* Його зсув відносно початку файлу зберігається за адресою 0x3C в DOS-заголовку.
* Знайдене значення 80 00 00 00 (позначено чорним) в форматі little-endian відповідає зсуву 0x00000080 (128 байтів).
* За цим зсувом знаходиться сам PE-заголовок, перші 24 байти якого є стандартними. Він починається з сигнатури 50 45 00 00 ("PE\0\0") і позначений темно-червоним.

Далі було визначено розмір **Optional Header**:

* Значення E0 00 (позначено фіолетовим), знайдене після основного PE-заголовка, в форматі little-endian дорівнює 0x00E0 (224 байти).

Одразу після Optional Header розташовані **заголовки секцій (SECTION\_HEADERS)** (виділені зеленим на другому скріншоті):

* Кількість секцій визначається з поля в PE-заголовку: 02 00 (позначено жовтим) означає 2 секції.
* Розмір одного заголовка секції становить 40 байтів.
* Загальний розмір усіх заголовків секцій розраховується як Кількість секцій \* 40 байтів, що в даному випадку становить 2 \* 40 = 80 байтів.

7. Відповідно до опису секцій скласти таблицю, в яку занести параметри свого файлу, вказані в розділах 3.3.1, 3.4.1 і 4 (перша таблиця)

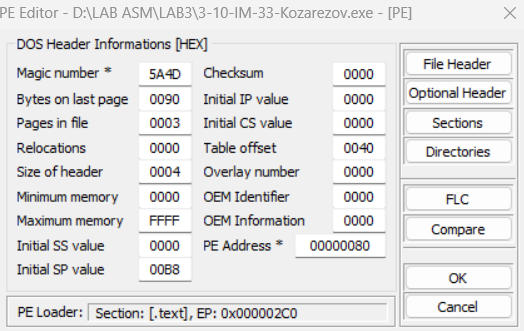
8. У останньому стовпчику таблиці розшифрувати виписані значення полів заголовка файлу. Таблицю привести в звіті по лабораторній роботі

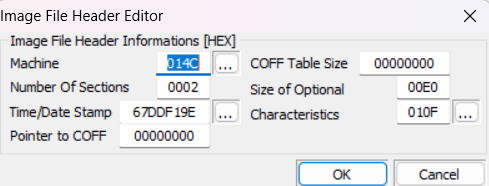
|  |  |  |
| --- | --- | --- |
| Machine Types | | |
| Поле | Значення поля | Опис поля |
| Machine type | 01 4С | Тип машини (CPU): Intel 386 або пізніше або схожі процесори |

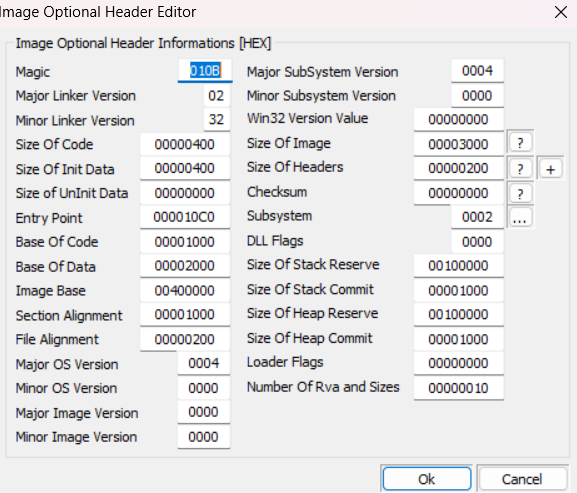
|  |  |  |
| --- | --- | --- |
| Optional Header Standard Fields (Image Only) | | |
| Поле | Значення | Опис |
| Magic | 01 0В | Число типу unsigned int, що ідентифікує стан файлу.  **0x10B** – ідентифікатор звичайного файлу **.exe**. |
| MajorLinkerVersion | 02 | Номер основної версії лінкера |
| MinorLinkerVersion | 32 | Номер другорядної версії лінкера. |
| SizeOfCode | 00 00 04 00 | Розмір секції коду |
| SizeOfInitializedData | 00 00 06 00 | Розмір ініціалізованого розділу даних |
| SizeOfUninitializedData | 00 00 00 00 | Розмір розділу неініціалізованих даних |
| AddressOfEntryPoint | 00 00 11 16 | Початкова адреса програми |
| BaseOfCode | 00 00 10 00 | Адреса початку секції коду |
| BaseOfData | 00 00 20 00 | Адреса початку секції змінних |

|  |  |  |
| --- | --- | --- |
| Section Table (Section Headers) | | |
| Поле | Значення | Опис |
| Name | .data | Назва секції: 8-байтовий рядок ASCII |
| VirtualSize | 00 00 04 8C | Загальний розмір секції |
| VirtualAddress | 00 00 20 00 | Адреса першого байту секції |
| SizeOfRawData | 00 00 04 00 | Розмір секції |
| PointerToRawData | 00 00 06 00 | Вказівник на першу сторінку секції в COFF файл |
| PointerToRelocations | 00 00 00 00 | Вказівник до початку входжень релокацій в секцію (для файлів .**ехе** рівний 0) |
| PointerToLinenumbers | 00 00 00 00 | Вказівник файлу на початок записів номерів рядків для секції |
| NumberOfRelocations | 00 00 | Кількість релокаційних входжень в секцію (для файлів .ехе рівний 0) |
| NumberOfLinenumbers | 00 00 | Кількість номерних записів секції |
| Characteristics | C0 00 00 40 | Прапори опису характеристики секції |

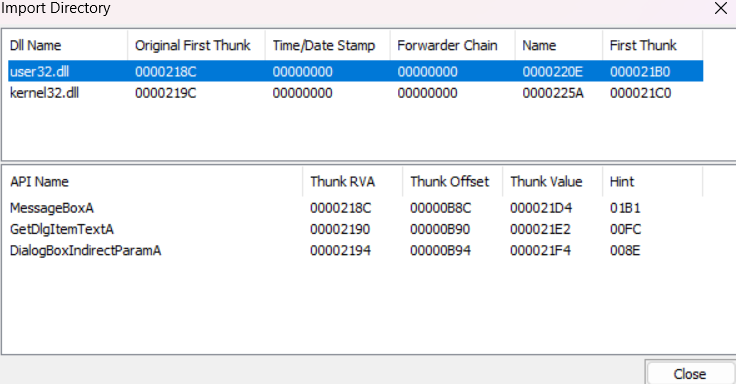
9. Провести дослідження того ж файлу за допомогою меню "**PE Editor**" безкоштовної програми **PE Tools**. Всі скріншоти вікон програми з даними, відповідними раніше побудованій таблиці, привести в звіті по лабораторній роботі.

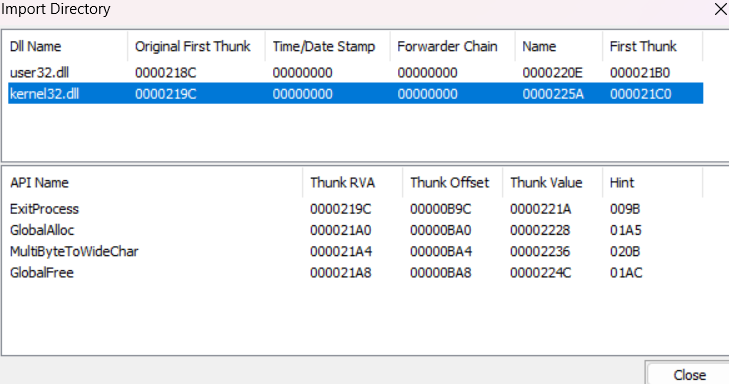






10. Дослідити таблицю імпорту (**Import Directory**) даного файлу і визначити, які саме функції використовуються з бібліотек, що підключаються. Скріншоти вікон **Import Directory** з функціями, що імпортуються, з кожного бібліотечного файлу привести в звіті по лабораторній роботі.





11. Знайти в тексті файлу по зсуву, узятому з побудованої таблиці, секцію з даними і переконатися, що текст оригінала пароля, що міститься в тексті програми, може бути легко виявлений за допомогою **HEX**-редактора. Привести скріншот цього фрагмента програми у вигляді **HEX** - коду в звіті по лабораторній роботі.

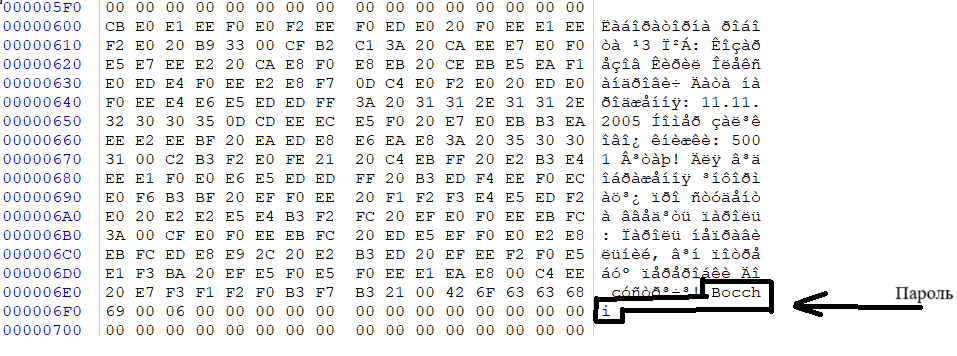
**Password**

Легкість, з якою пароль знаходиться через HEX-редактор (як показано на скріншоті), є абсолютно неприпустимою практикою в реальній розробці. Це свідчить про повну відсутність захисту від потенційних зловмисників.

12. Виконати шифрування пароля за допомогою функції **XOR**, знову скомпілювати **EXE** -файл і переконатися, що тепер вони не виявляються явним чином в тексті виконуваного **EXE** - файлу. Привести скріншоти цієї програми в режимах «**Hex**» і «**Text**» в звіті по лабораторній роботі.

**Шифрування паролю функцією XOR**

Зашифруємо поточний пароль для відображення даних «**Bocchi**» за ключем «**!@!@!!**»:



B (01000010) XOR ! (00100001) = c (01100011)

o (01101111) XOR @ (01000000) = / (00101111)

c (01100011) XOR ! (00100001) = B (01000010)

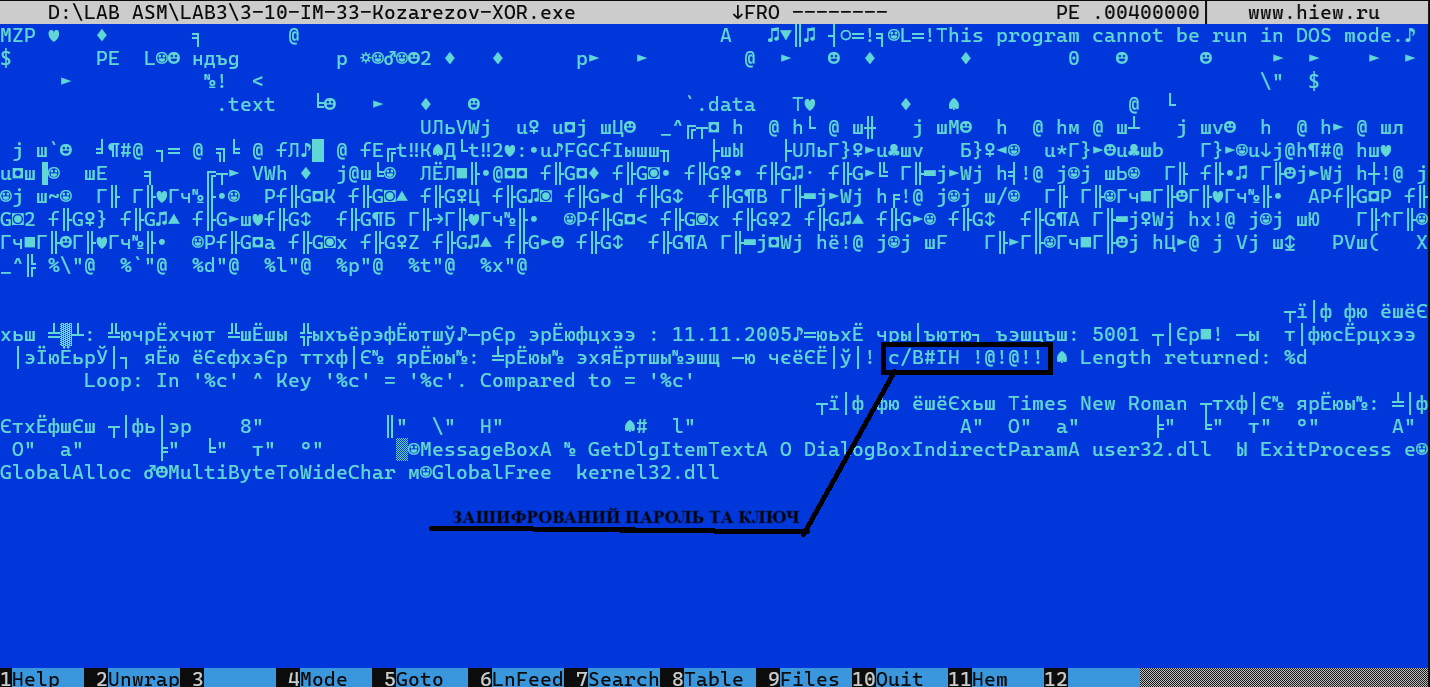
c (01100011) XOR @ (01000000) = # (00100011)

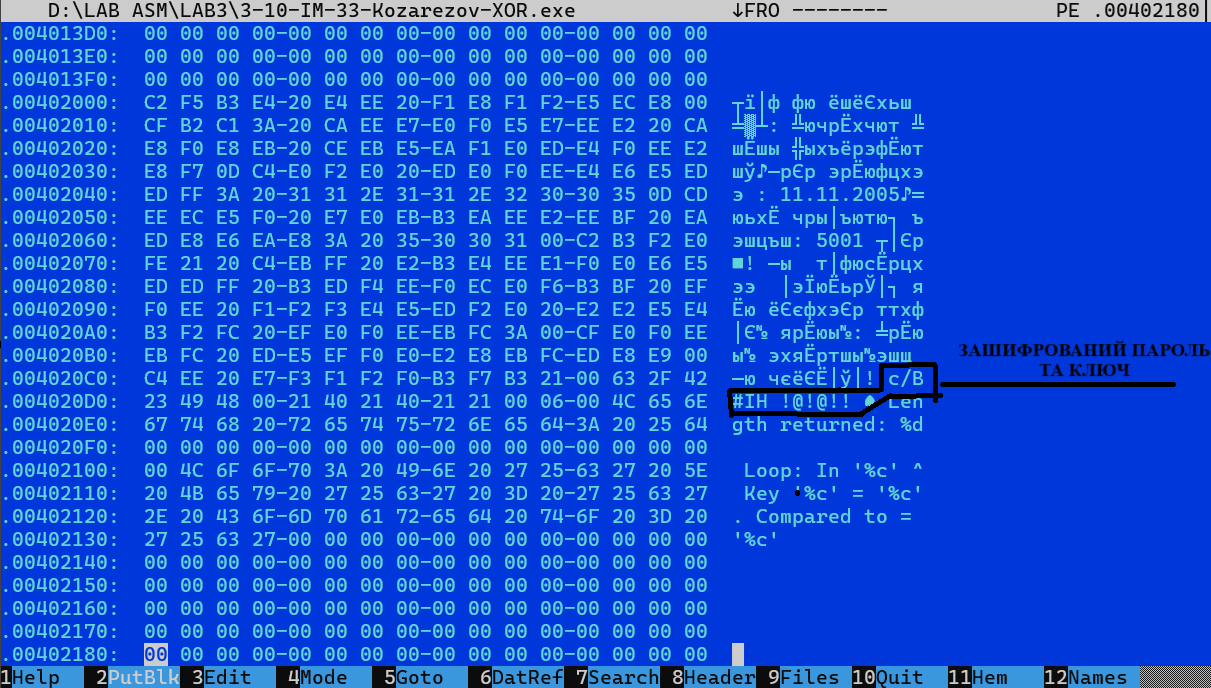
h (01101000) XOR ! (00100001) = I (01001001)

i (01101001) XOR ! (00100001) = H (01001000)

Після шифрування ми отримали зашифрований пароль c/B#IH

Зашифрований пароль у програмі:





Як демонструють скріншоти, і пароль, і ключ все ще можна виявити у секції даних. Однак, тепер пароль представлений неявно, і не кожен зможе зрозуміти, що це саме він. Також, для розшифрування потрібен ключ, пошук якого є непростим завданням. Тому можна стверджувати, що захищеність зросла

13. Зробити висновки по лабораторній роботі.

**Висновки**

У рамках виконання лабораторної роботи №3 було здобуто практичний досвід аналізу виконуваних файлів формату EXE. Освоєно методику роботи із заголовками DOS та PE, включаючи їх компонентний аналіз. Вивчено структуру та локалізацію заголовків секцій (Section Headers). Набуто навичок визначення віртуальних адрес початку секцій, що є необхідним для пошуку потенційно чутливих даних, зокрема паролів; коректність визначення підтверджено порівняльним аналізом у програмах HIEW32 та PE-Tools. Окремо досліджено метод XOR-шифрування шляхом реалізації двох програмних варіантів: без шифрування пароля та з його XOR-шифруванням за допомогою відкритого (явно заданого) ключа. Зроблено висновок про неприпустимість зберігання паролів у незашифрованому вигляді через легкість їх виявлення спеціалізованим ПЗ (HIEW32, WinHex). Водночас встановлено, що навіть XOR-шифрування з відкритим ключем забезпечує лише обмежений рівень захисту, оскільки існує можливість локалізації ключа та шифротексту. Це підкреслює важливість застосування криптографічно стійкіших методів, таких як шифрування із закритим ключем.