**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-21

Лесько Дмитро Миколайович

номер у списку групи - 14

**Київ 2024**

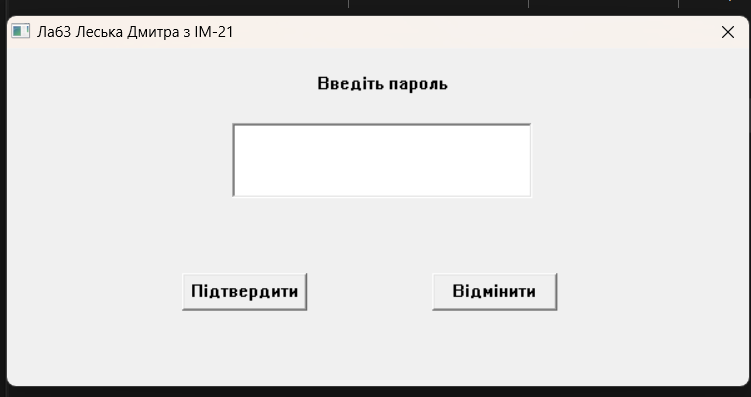
**Мета роботи:** ознайомлення із специфікацією COFF та вивчення прийомів дослідження структури файлу РЕ-формату.

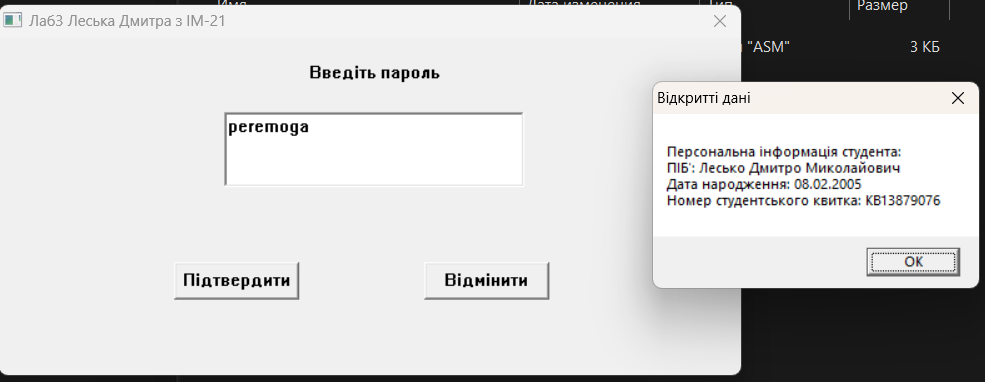
**Структура програм формату ЕХЕ:**

Файл з розширенням .exe є виконуваним файлом, який використовується у системах DOS, Microsoft Windows та інших операційних системах. Він містить дві основні частини: заголовок, який містить інформацію для завантажувача програми, і модуль завантаження. Перші 25 рядків файлу складаються з заголовка, який включає дані про розмір файлу та обсяг пам'яті, необхідний для його виконання.

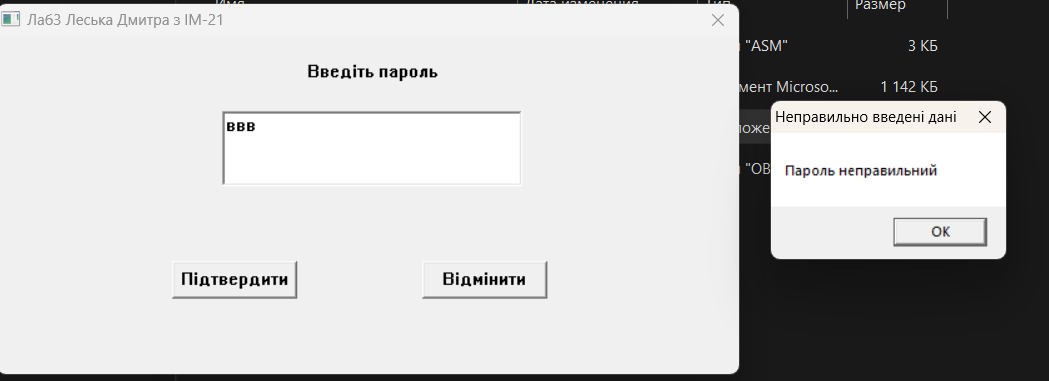
Після заголовка йде таблиця налаштування адрес, що містить пари значень - зміщення та відповідну адресу, які вказують, куди в пам'яті потрібно завантажити конкретні частини програми.

**Скріншоти роботи програми:**

Після компіляції програми до стану executable файлу, лабораторна робота виводить діалогове вікно:  


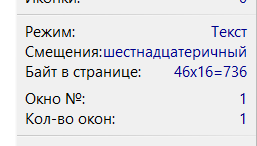
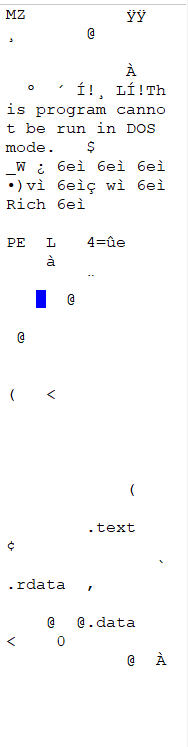
Далі програма пропонує ввести пароль, щоб отримати доступ персональних даних студента. Якщо ввести правильний пароль, то вона виконає відповідну функцію і вийде з циклу виконання програми:  


Якщо ж ввести неправильний пароль, то програма виведе в новому меседж боксі повідомлення про помилку і завершить програму:

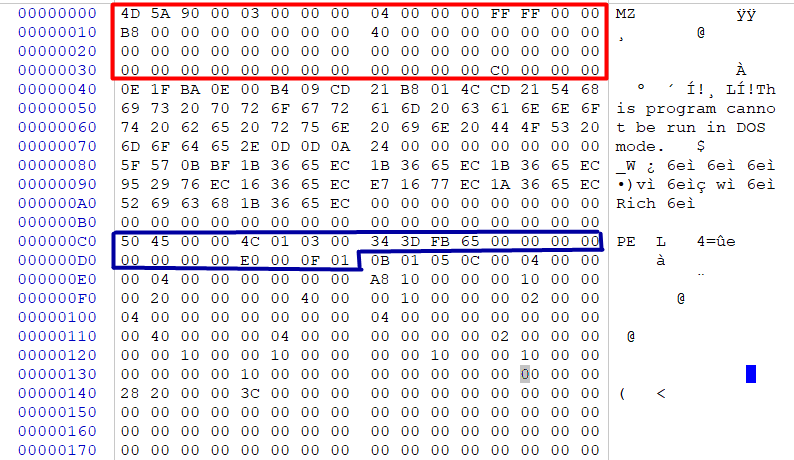


**Дослідження файлу за допомогою програми WinHEX:**

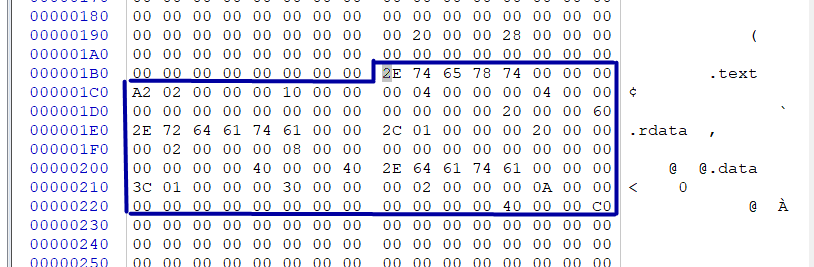
Текстовий формат:



Далі відкриваємо шістнадцятковий формат. Червоним кольором я обвів область MS-DOS заголовка, а синім- PE Header:



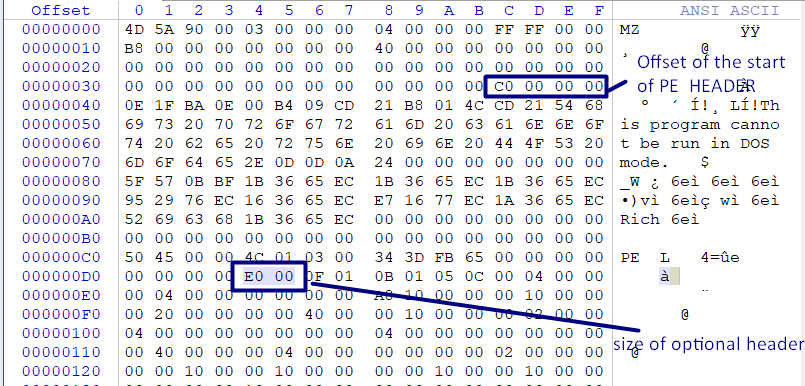
І також підкреслив SECTION\_HEADER:



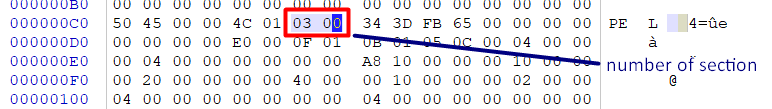
**Пояснення обведення секцій:**Отже, ми обвели секції, які були вказані у завдання. Чому ми обвели саме їх:  
1. Перше обведення- DOS\_HEADER, він знаходиться на початку і стало складається з 64 байтів (на першій фотографії дослідження шістнадцяткового коду він виділений синім)

2. Після визначення першого хедера, по стандартній методиці шукаємо зсув 0x3C, за яким далі визначаємо зсув початку PE\_HEADER, який займає 24 байти (виділений червоним на тій же фотографії, що й DOS\_HEADER).

3. Додатково можемо легко визначити у PE\_HEADER розмір опційного заголовку:  
Знаходимо у PE\_HEADER SizeOfOptionalHeader: E000, далі за допомогою обернення бітів визначаємо його розмір: 000Eh = 224 байти



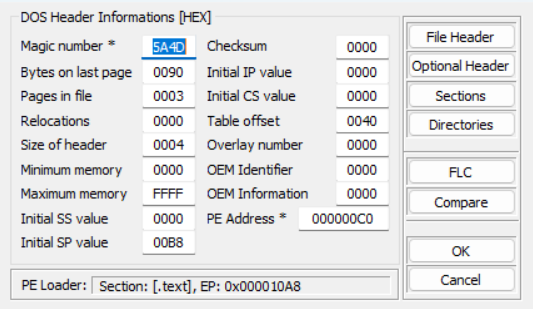
4. Після PE\_HEADER починаємо відлік і знаходимо SECTION\_HEADER (виділений на синім на наступній фотографії після DOS i PE HEADERS). Рахуємо його розмір:  
розмір секції- 40 байтів, у PE\_HEADER вказано їх кількість: 0300, тобто три секції. Відповідно розмір SECTION\_HEADER = 4 \* 30 = 120 байтів.

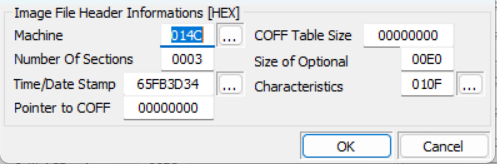


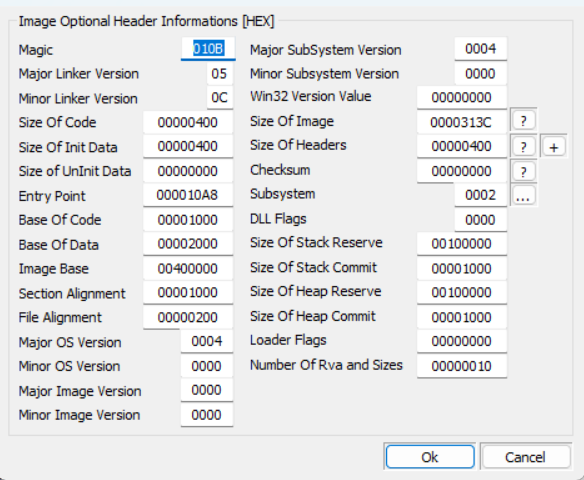
**Таблиця параметрів файлу:**

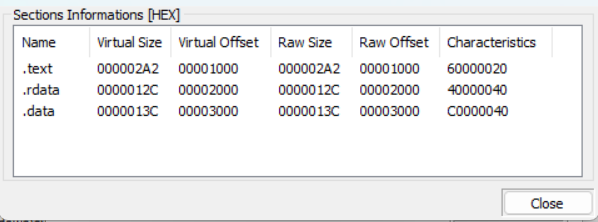
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поле** | | | **Значення** | | **Опис поля** | | **Опис значення поля** |
| **Machine type** | | | 014С | | Тип машини (CPU) | | Intel 386 або пізніша, та інші схожі процесори |
| Optional Header Standard Fields | | | | | | |
| **Magic** | 010B | | Число типу unsigned integer, ідентифікує стан файлу | | Ідентифікатор нормального файлу типу .ехе | |
| **Major linker version** | 05 | | Номер версії основного лінкера | | | |
| **Minor linker version** | 0С | | Номер версії меншого лінкера | | | |
| **Size Of Code** | 00000400 | | Розмір секції коду | | | |
| **Size Of Initialized Data** | 00000400 | | Розмір секції ініціалізованих змінних | | | |
| **Size Of Uninitialized Data** | 00000000 | | Розмір секції неініціалізованих змінних | | | |
| **Address Of Entry Point** | 000010А8 | | Адреса вхідної точки в програму, початкова адреса програми | | | |
| **Base Of Code** | 00001000 | | Адреса початку секції коду при завантаженні в пам’ять | | | |
| **Base Of Data** | 00002000 | | Адреса початку секції змінних при завантаженні в пам’ять | | | |
| Section Table (Section Headers) | | | | | | |
| **Name** | .text | | Назва секції - 8 бітний рядок ASCII | | | |
| **Virtual Size** | 000002A2 | | Розмір секції, завантаженої в пам’ять | | | |
| **Virtual Address** | 00001000 | | Адреса першого байту секції | | | |
| **Size Of Raw**  **Data** | 00000400 | | Розмір секції | | | |
| **Pointer To**  **Raw data** | 00004000 | | Пойнтер до першої сторінки секції у файлі COFF | | | |
| **Pointer to relocation** | 00000000 | | Пойнтер до початку входжень релокацій в секцію (для файлів .ехе рівний 0) | | | |
| **Pointer to Linenumbers** | 00000000 | | Пойнтер до початку входжень в секцію чисел | | | |
| **Number of relocation** | 0000 | | Кількість релокаційних входжень в секцію (для файлів .ехе рівний 0) | | | |
| **Number of Linenumbers** | 0000 | | Кількість числових входжень у секцію | | | |
| **Characteristics** | 60000020 | | Прапори опису характеристики секції | | | |

**Дослідження в PE Tools:**







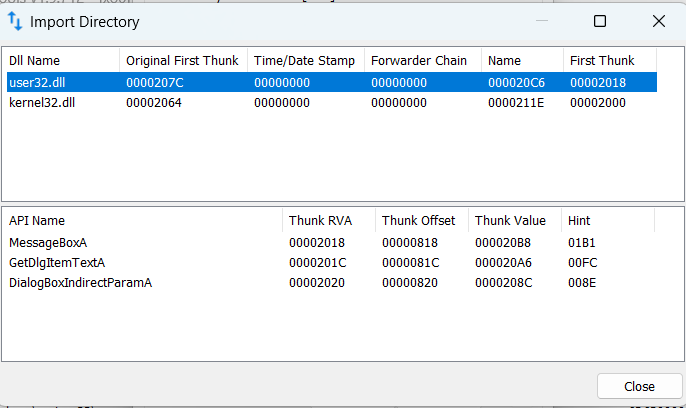


За допомогою програми PE Tools бачимо, що наші дані, які ми занесли в таблицю за допомогою прописаних відступів у хедерах, які були вказані у вкладеній книзі, співпадають з даними, які нам автоматично вивела програма.

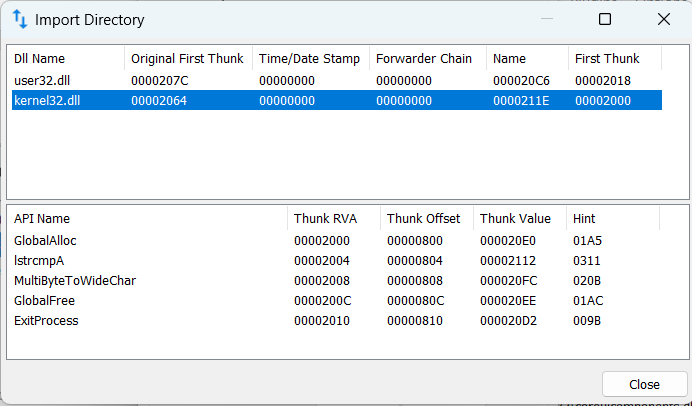
**Дослідження таблиці імпорту:**

У програмі використовуються дві бібліотеки: user32.dll та kernel32.dll. Нижче наведено список функцій цих бібліотек, які використовуються в програмі і який було знайдено за допомогою PE Tools.

Імпорти для бібліотеки user32.dll

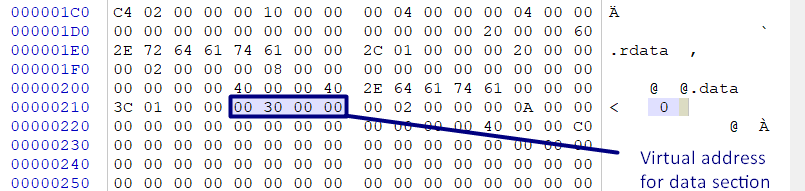


Імпорти для бібліотеки kernel32.dll

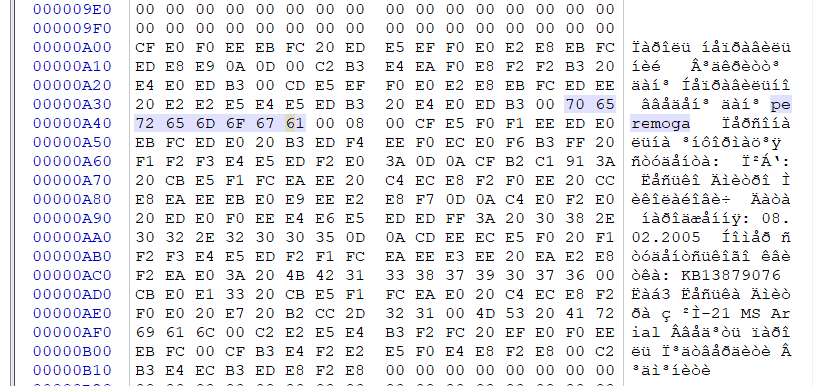


**Виявлення паролю у тексті файлу за допомогою HEX редактору:**

Щоб знайти зсув початку секції data, де зберігається пароль, можна додати Virtual Address секції data, що було знайдено у Section.



Отримано 003000. Отже секція data починається за адресою 00403000. Як бачимо, за допомогою відступів від заголовків та мануального пошуку в текстовому форматі нашого файлу ми з легкістю знайшли текст нашого паролю, що робить нашу програму дуже незахищеною:



**Дослідження файлу, де пароль зашифрований методом XOR:**

Для початку давайте зашифруємо пароль вручну за допомогою алгоритму шифрування XOR:

Ключ: \*

Пароль: peremoga

Переведення символів у двійкову систему координат використовуючи ASCII таблицю:

\* = 4210 = 001010102

p = 11210 = 011100002

e = 10110 = 011001012

r = 11410 = 011100102

e = 10110 = 011001012

m = 10910 = 011011012

o = 11110 = 011011112

g = 10310 = 011001112

a = 9710 = 0110000122

**Шифрування:**

p XOR \* = 011100002 XOR 001010102 = 010110102 = 9010 = Z

e XOR \* = 011001012 XOR 001010102 = 010011112 = 7910 = O

r XOR \* = 011100102 XOR 001010102 = 010110002 = 8810 = X

e XOR \* = 011001012 XOR 001010102 = 010011112 = 7910 = O

m XOR \* = 011011012 XOR 001010102 = 010001112 = 7110 = G

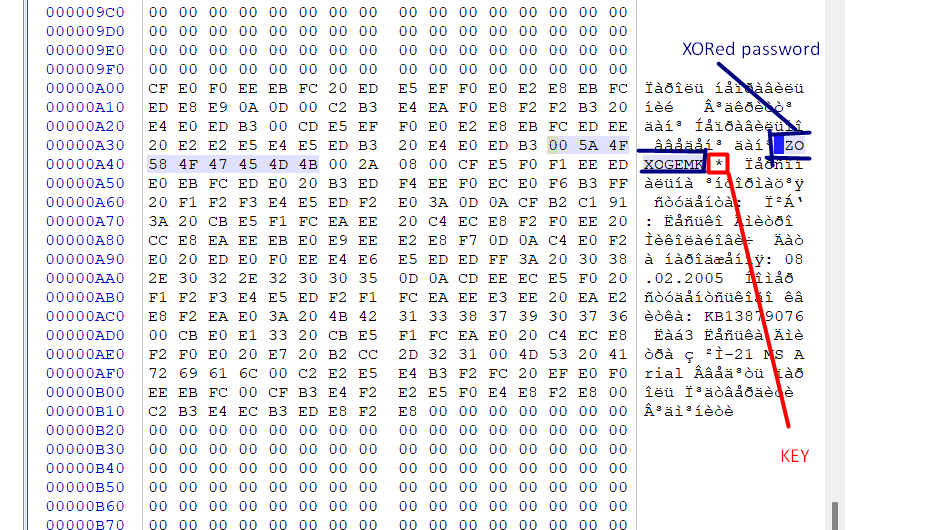
o XOR \* = 011011112 XOR 001010102 = 010001012 = 6910 = E

g XOR \* = 011001112 XOR 001010102 = 010011012 = 7710 = M

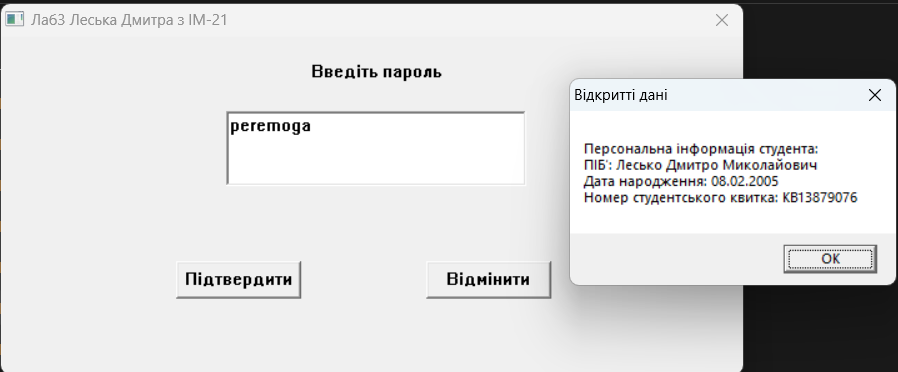
a XOR \* = 011000012 XOR 001010102 = 010010112 = 7510 = K

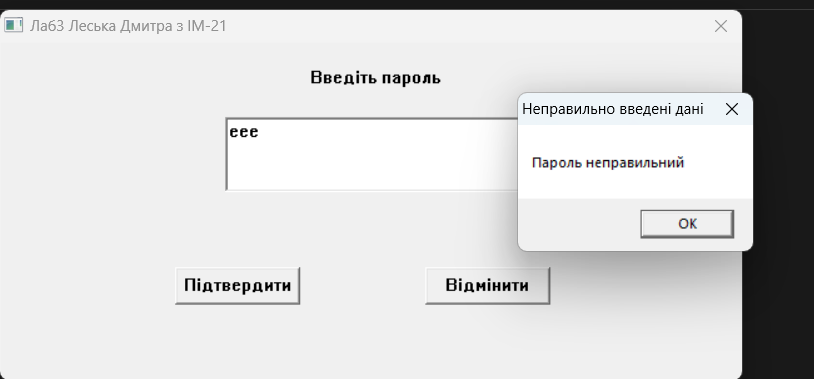
Зашифрований пароль- **ZOXOGEMK**

Ми склали нову програму, новий алгоритм перевірки паролю, вона працює так само, проте тепер пароль у програмі знаходиться у зашифрованому вигляді і тепер його складніше розшифрувати на пряму.









Тобто ми отримало повністю, той самий бажаний результат: пароль зашифрований, перевіряється правильно, тепер його не так легко знайти звичайному користувачу, хоча всі інші дані не є зашифрованими.

**Висновок:**

Під час виконання лабораторної роботи я вивчав структуру виконуваних файлів з розширенням .exe, які використовуються в операційних системах DOS і Microsoft Windows для запуску програм. Глибокіше розібравшись у цій структурі, я зосередився на двох основних складових частинах файлу: заголовку, що містить інформацію для завантажувача програми, і модулі завантаження.

Я також ретельно дослідив відступи в hex програмах. Відступи, як правило, вказують на рівень вкладеності і структуру програмного коду. Вони є важливим елементом для зручності читання та розуміння програми, особливо коли мова програмування не має явних конструкцій блокування коду

У процесі вивчення, я розглянув різні параметри файлу і методи їх знаходження та зміни, використовуючи інструменти, такі як PE Tools. Окрім того, я використовував цю програму для аналізу імпортованих бібліотек і функцій, які використовуються в файлі, що дозволило краще розібратися у внутрішній структурі програми.

Під час створення двох програм, головною метою було забезпечити безпеку паролю. У першій програмі пароль було використано прямо в коді, що створює потенційні ризики безпеки. У другій програмі, я використовував метод шифрування XOR для захисту паролю. У коді цієї програми міститься лише зашифрований пароль і ключ, що робить його більш безпечним у порівнянні з першим варіантом.