**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

Лабораторна робота №3

з дисципліни

«Системне програмування»

на тему

«Дослідження структури програм формату EXE»

Виконала: Перевірив:

Студентка групи ІМ-21 доцент Павлов В. Г.

Дворецька Анастасія Вадимівна

Номер у списку групи: 6

**Київ 2024**

Варіант 6

**Мета:**

Ознайомлення із специфікацією COFF (Common Object File Format). Вивчення прийомів дослідження структури файлів PEформату.

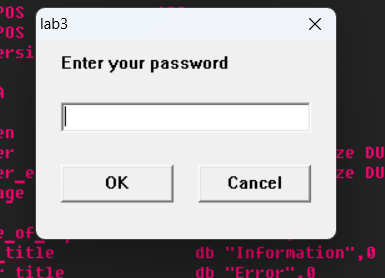
**1.Вивчена структура програм формату EXE:**

Я зрозуміла, що ехе файл починається з Dos заголовку, який займає 64 байти,останні 4 з яких займає посилання на PE заголовок, далі йде Dos заглушка,після неї PE заголовок,який починається з сигнатури, після нього йде необов’язковий заголовок. Після нього вже SECTION\_HEADERS.

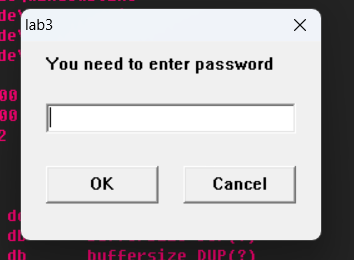
Секції програми розташовані слоями на певних частинах пам’яті,у моєму випадку 3 секції: секція з кодом(.code),з даними(.data) і з таблицею імпортів(.idata),ці секції розташовані в самому кінці ехе файлу.Перед ними якраз і розташовані заголовки,які вказують як правильно відобразити їх у пам’яті.

**2.Скріншоти виконання програми:**

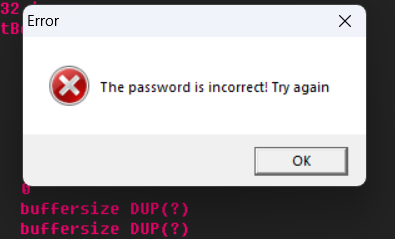
Відразу після запуску:



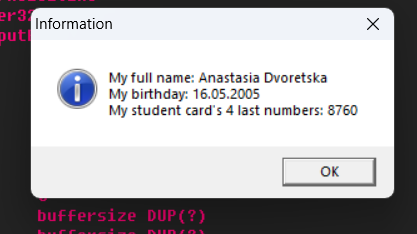
Якщо нічого не ввести:



Якщо ввести неправильний пароль:

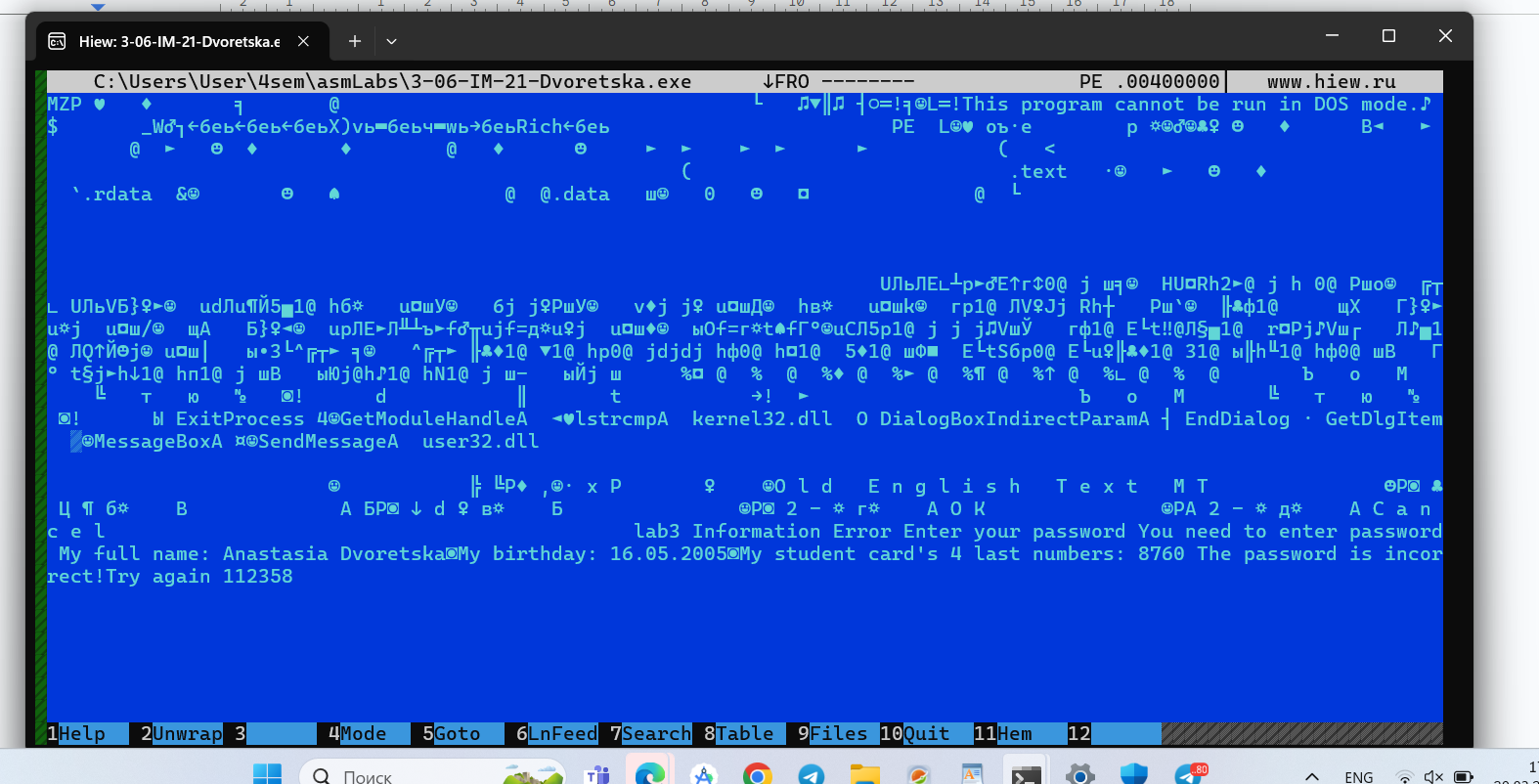


Якщо ввести правильний пароль:

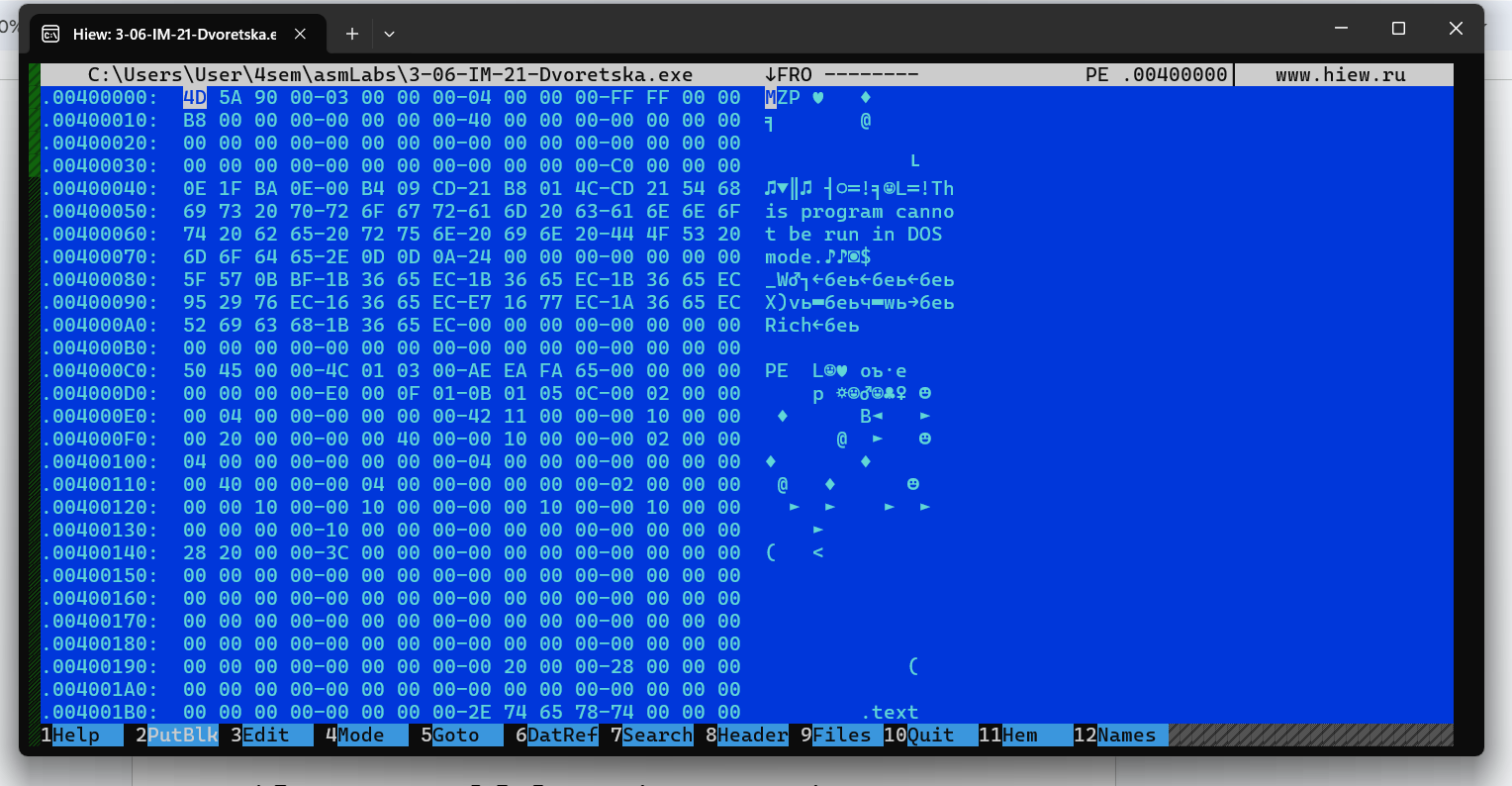


**3.Дослідження файлу ехе за допомогою hiew32:**

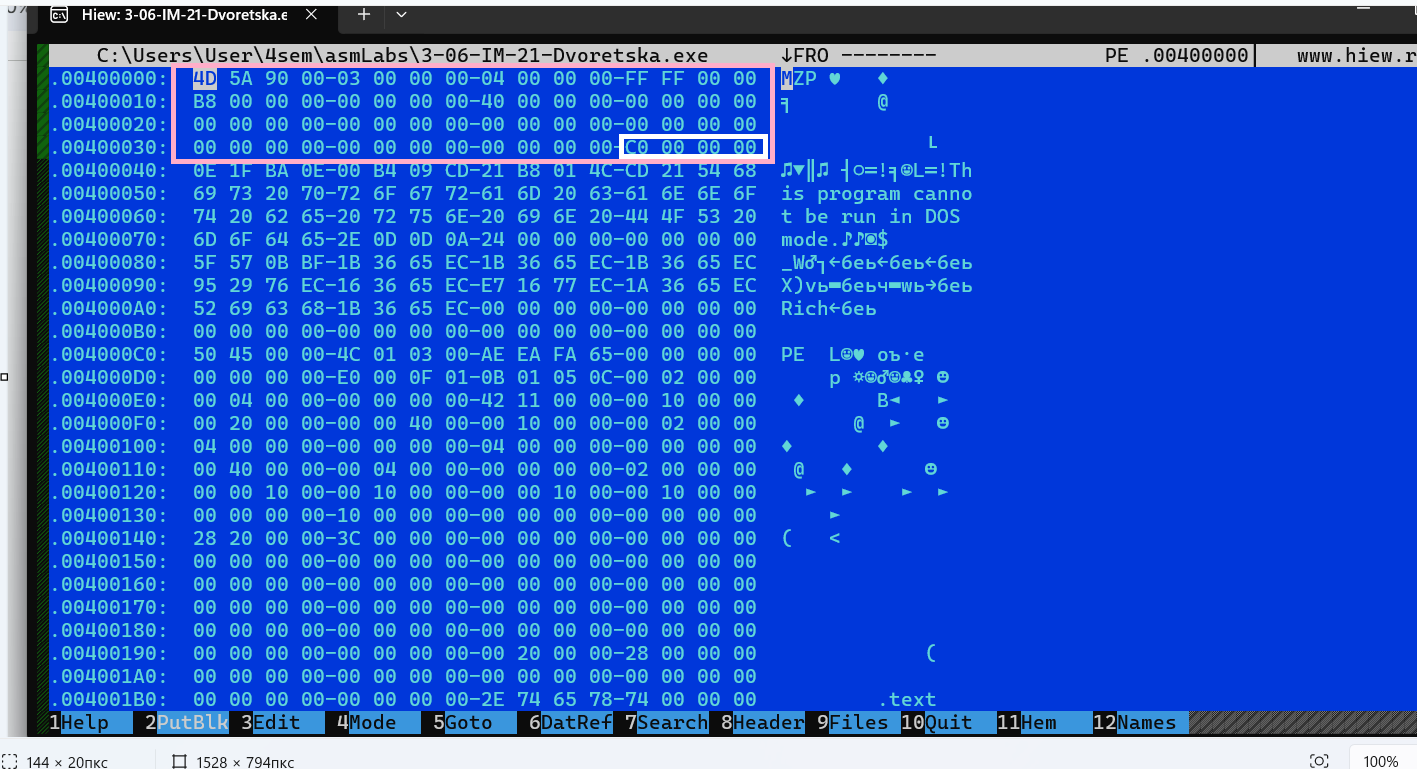
Вивід у текстовому форматі:

****

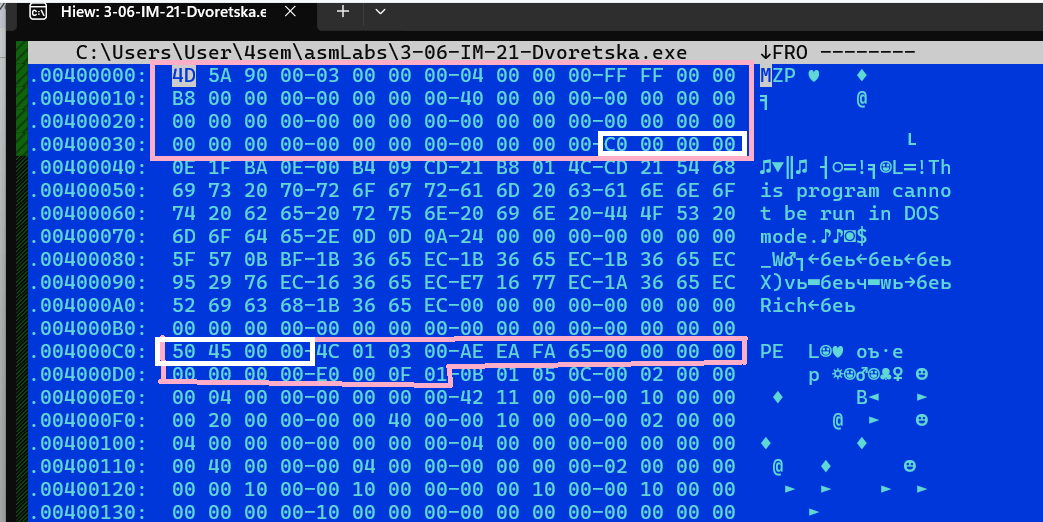
Вивід у 16-ковому:



DOS\_HEADER займає перші 64 байти файлу, останні 4 байти вказують на зсув від початку,на якому розташован PE\_HEADER

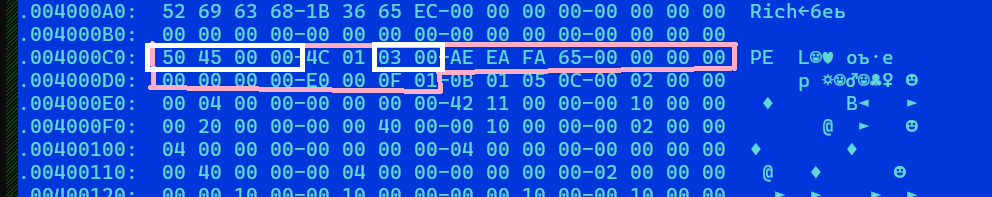


PE\_HEADER починається з сигнатури,яка займає 4 байти,а сам заголовок займає 20 байт

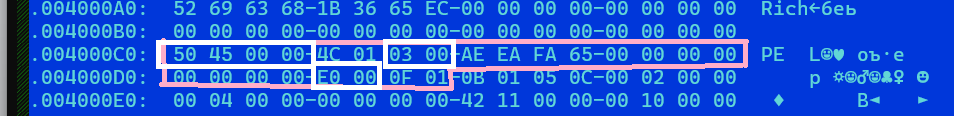


Корисні дані,які я витягла:

* другі два байти PE\_HEADER відповідають кількості секцій у моєму файлі, у мене,як це можна побачити їх 3.

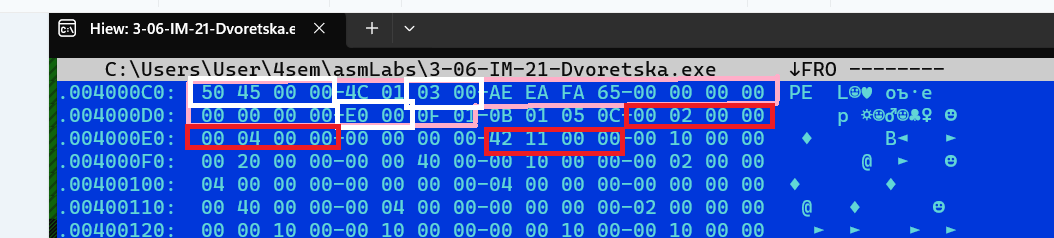


* передостанні два байти визначають розмір необов’язкового заголовку,вони записані у зворотньому порядку,тож розмір: 00 Е0, що дорівнює 224 байтам.



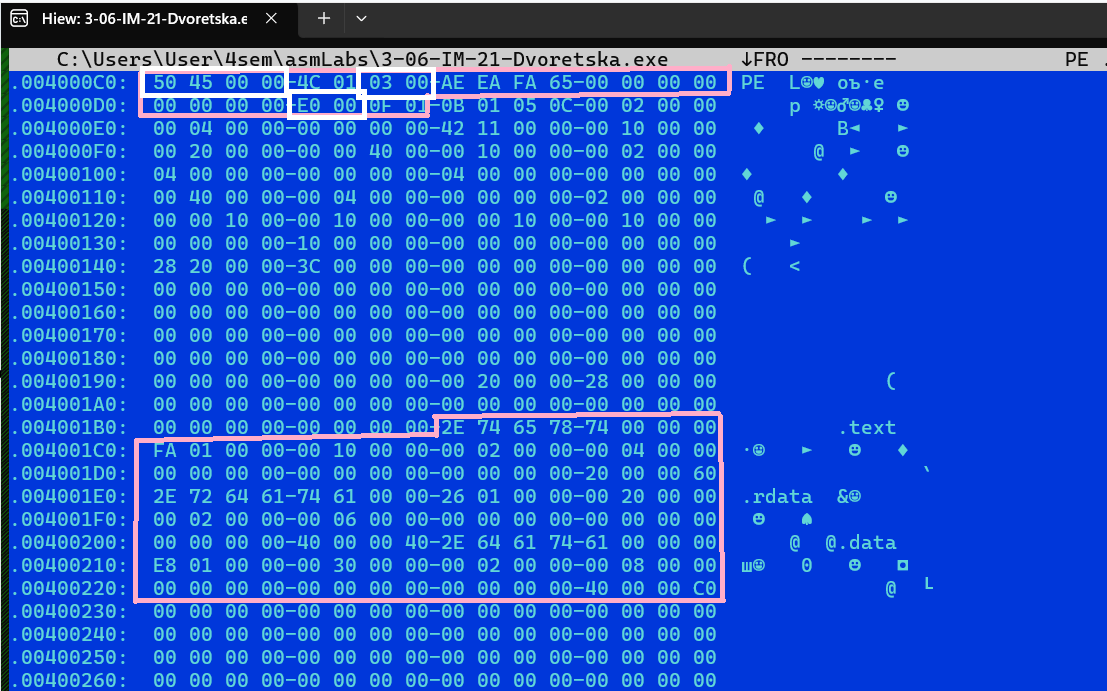
Хочу також трошки пояснити звідки я взяла дані у таблиці : перші два байти відповідають сигнатурі заголовку,далі два байти на версію компанувальника,далі червоним я позначила :

1. 4 байти що відповідають розміру секції коду
2. 4 байти,що відповідають розміру секції ініціалізованих даних
3. 4 байти,що відповідають точці запуску програми



Далі по порядку йдenm baseOfCode, baseOfData, imageBase, sectionAlignment, fileAlignment

Відрахувавши з кінця PE\_HEADER 224 байти,які дорівнюють рівно 14 рядків отримала розташування SECTION\_HEADERS, розмір якого рахується N\*40 байт,як я раніше з’ясувала,N = 3, тож 3 \* 40 = 120 байт.

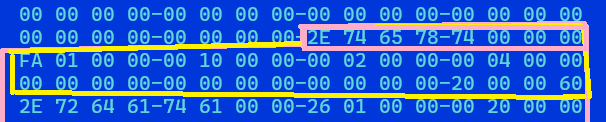


Хочу ще подрібно розписати як саме я заповнила останню частину таблиці Section Headers, я її заповнювала на прикладі сегменту коду,отже:

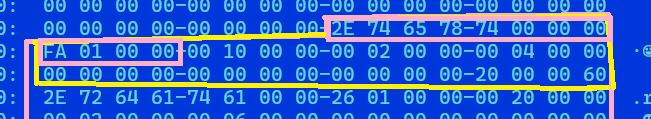
* секція з кодом розташована першою в SECTION\_HEADERS,тож їй належать перші 40 байтів



* перші 8 байт відповідають за ім’я секції:



* далі 4 байти відповідають за розмір секції в пам’яті:



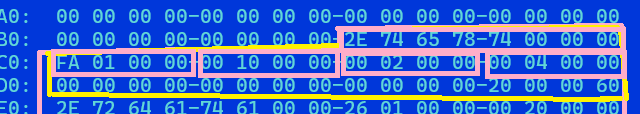
* наступні 4 байти- RVA в пам’яті:



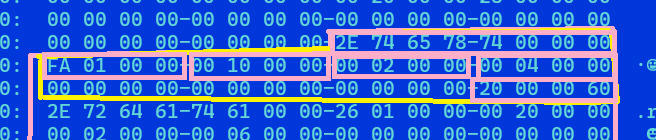
* далі 4 байти - розмір секції в ехе файлі:



* 4 байти відповідають за зсув у файлі до початку даних секцій:



* і останні 4 байти належать характеристикам секції:

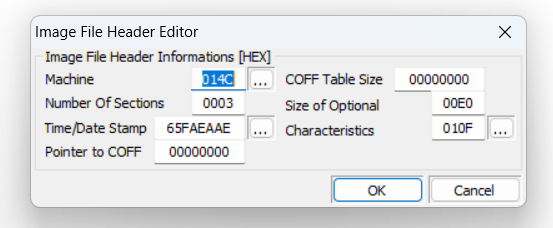


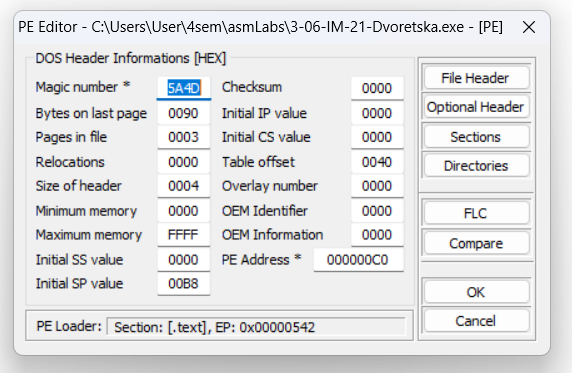
**4. Відповідно до опису секцій [1] скласти таблицю, в яку занести параметри свого файлу, вказані в розділах 3.3.1, 3.4.1 і 4 (перша таблиця).**

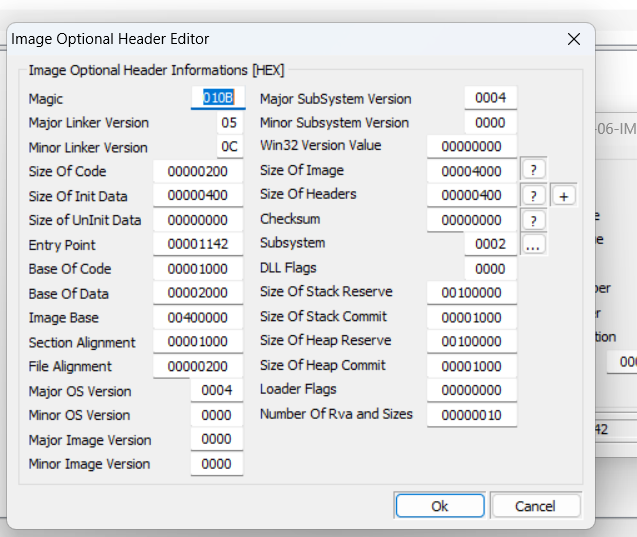
| **3.3.1(Machine Types)** | | | |
| --- | --- | --- | --- |
| Поле | Значення | Опис | |
| Machine type | 01 4С | Перші два байти PE\_HEADER, відповідають за тип машини(позначення процесору).Intel 386 або пізніше | |
| **3.4.1(Optional Header Standard Fields)** | | | |
| magic | 01 0В | формат,що показує для якої розрядності була написана програма | |
| majorLinkerVersion | 05 | визначають версію компонувальника | основного |
| minorLinkerVersion | 0С | меншого |
| sizeOfCode | 00 00 02 00 | розмір секції коду(кратен вирівнюванню в ехе) | |
| sizeOfInitializedData | 00 00 04 00 | розмір секції ініціалізованих даних(також кратен вирівнюванню) | |
| sizeOfUninitializedData | 00 00 00 00 | розмір секції неініціалізованих даних | |
| addressOfEntryPoint | 00 00 11 42 | RVA точки входу в програму | |
| baseOfCode | 00 00 10 00 | RVA секції з кодом | |
| baseOfData | 00 00 20 00 | RVA секції з даними | |
| imageBase | 00 40 00 00 | у нього записано інформацію про бажану адресу програми | |
| sectionAlignment | 00 00 10 00 | вирівнювання у віртуальній пам’яті, також є зміщенням між сегментами даних | |
| fileAlignment | 00 00 02 00 | вирівнювання в ехе файлі,розмір секцій кратен йому | |
| **4(Section Table/ Section Headers)** | | | |
| name | .text(2E 74 65 78 74 00 00 00) | ім’я секції | |
| virtualSize | 00 00 01 FA | розмір секції коду у віртуальній пам’яті | |
| virtualAddress | 00 00 10 00 | RVA адреса секції коду | |
| sizeOfRawData | 00 00 02 00 | розмір секції в ехе файлі | |
| pointerToRawData | 00 00 04 00 | RVA адреса в ехе файлі | |
| pointerToRelocations | 00 00 00 00 | у ехе файлах завжди дорівнюють 0 | |
| pointerToLineNumbers | 00 00 00 00 |
| numberOfRelocations | 00 00 00 00 |
| numberOfLineNumbers | 00 00 00 00 |
| characteristics | 60 00 00 20 | характеристики секції коду(секція доступна для читання та запису, має дозвіл на виконання, але не має захисту від запису.) | |

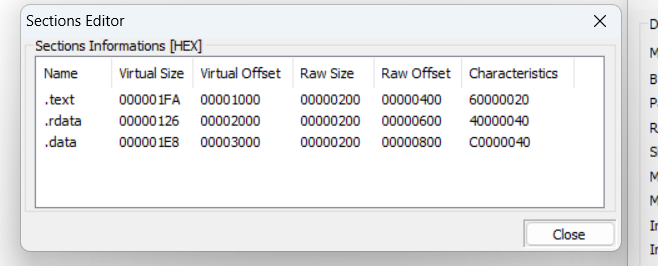
**5. Провести дослідження того ж файлу за допомогою меню "PE Editor":**

Провевши дослідження за допомогою PE EDITOR я зрозуміла,що правильно знайшла й записала у таблицю шукані дані,адже все співпало.

****

****

****

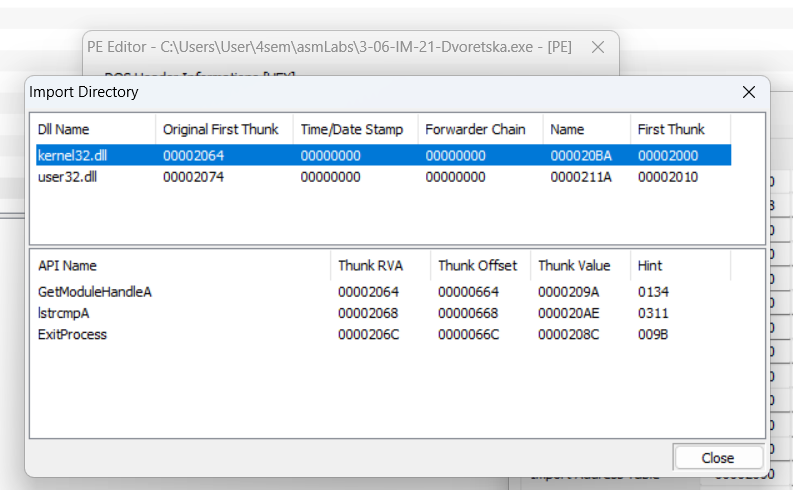
****

**6. Дослідити таблицю імпорту (Import Directory) даного файлу і визначити, які саме функції використовуються з бібліотек, що підключаються:**

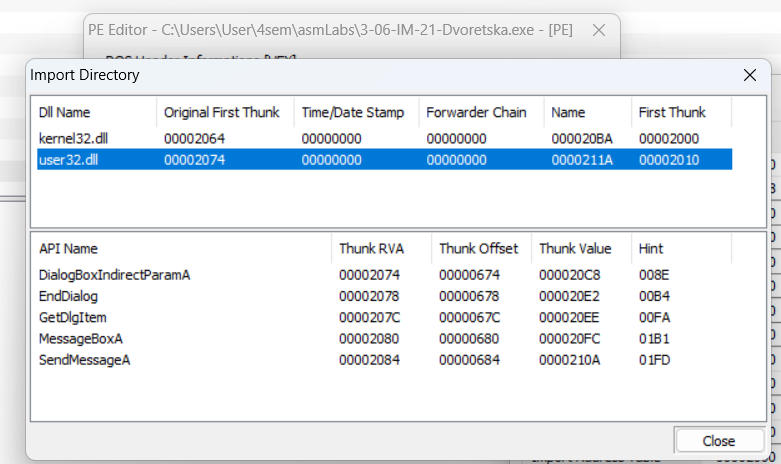
Для створення своєї програми я імпортувала дві бібліотеки:

kernel32.dll i user32.dll

* Ось функції,які використовую з бібліотеки kernel32.dll:

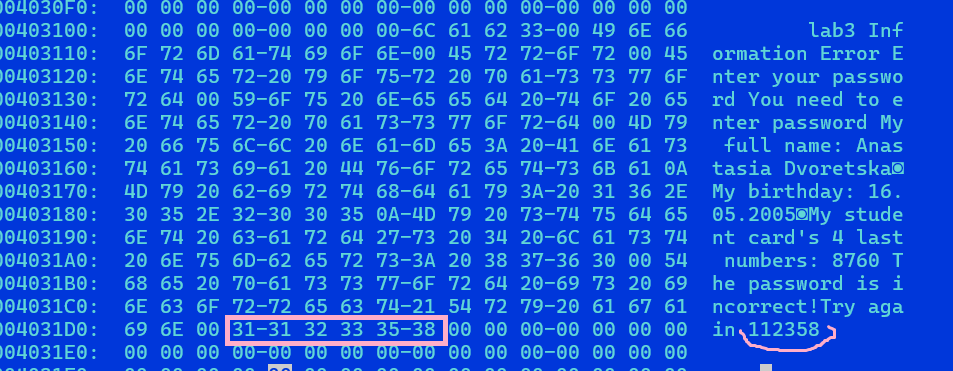


* І ось з бібліотеки user32.dll:



**7. 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:**

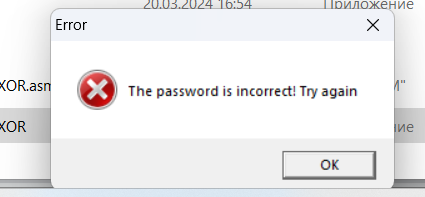
Дійсно,пароль дуже легко виявити за допомогою будь-якого НЕХ-редактора,що не є безпечним



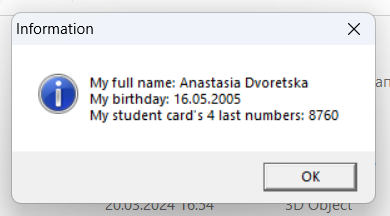
**8.Виконати шифрування пароля за допомогою функції XOR, знову скомпілювати EXE -файл і переконатися, що тепер вони не виявляються явним чином в тексті виконуваного EXE - файлу.**

Демонстрація роботи зашифрованої програми:

* Невірно введений пароль



* Вірно введений пароль



Мій пароль - 112358

Ключ для шифрування ХOR - xastia,

вони однакові за довжиною,що є доволі зручним

1 = 49 = 00110001 x = 120 = 01111000

1 = 49 = 00110001 a = 97 = 01100001

2 = 50 = 00110010 s = 115 = 01110011

3 = 51 = 00110011 t = 116 = 01110100

5 = 53 = 00110100 i = 105 = 01101001

8 = 56 = 00110111 a = 97 = 01100001

| Пароль | 00110001 | 00110001 | 00110010 | 00110011 | 00110100 | 00110111 |
| --- | --- | --- | --- | --- | --- | --- |
| Ключ | 01111000 | 01100001 | 01110011 | 01110100 | 01101001 | 01100001 |
| XOR | 01001001 | 01010000 | 01000001 | 01000111 | 01011101 | 01010110 |

Отже,пароль після шифрування :

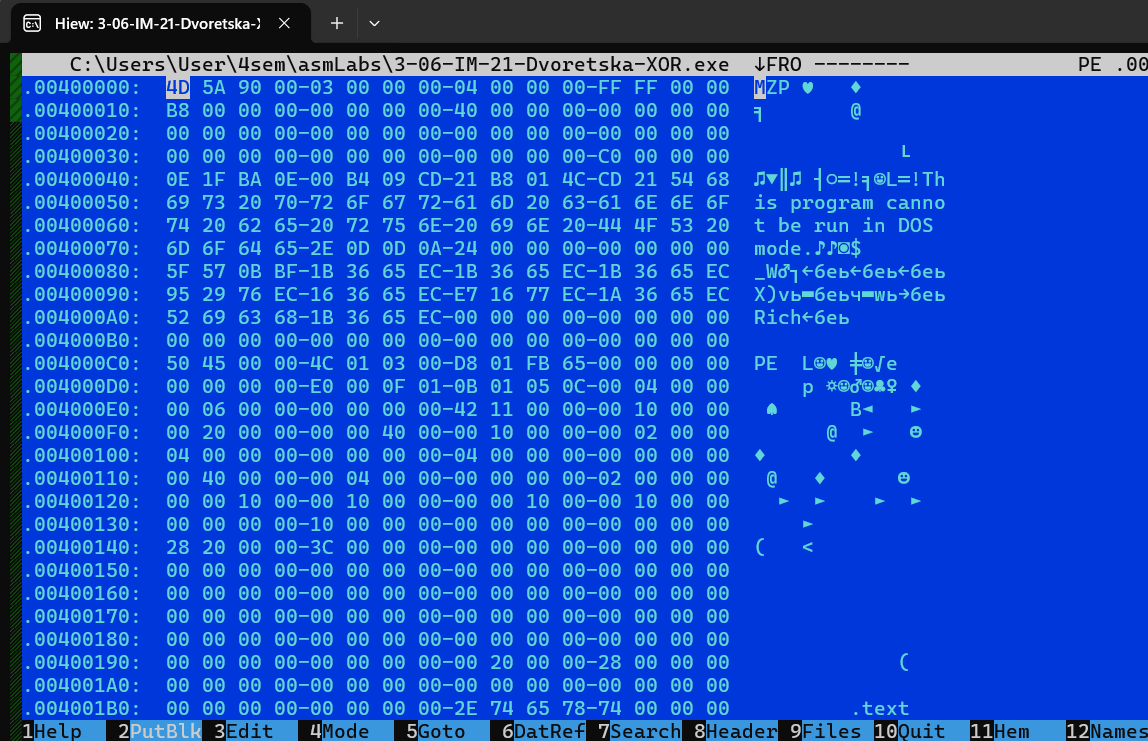
**IIJKM@**

Тепер,якщо відкрити програму у будь-якому НЕХ-редакторі,нерозшифрований видно не буде.Звичайно,що у межах лабораторної великого сенсу в цьому немає,адже інформація не зашифрована,да і ключ видно,тож якщо постаратись можна легко розшифрувати пароль,але це все ж таки дуже повчальний досвід,який змушує замислитись над безпекою даних.

Ось вигляд у текстовому форматі зашифрованого файлу:



Ось у 16-ковому:



І ось мій пароль



|  |  |
| --- | --- |
|  |  |

**9.Висновок:**

Протягом виконання лабораторної роботи я дізналась про структуру ехе файлів і глибоко її дослідила.Тепер я маю розуміння як мій компьютер розуміє,що робити,коли я в свою чергу просто відкриваю ехе файл.Я навчилась працювати з НЕХ-редактором hiew32 та PE TOOLS, знаю як з їх допомогою досліджувати структуру ехе файлів.Усі дані,які я обводила у hiew32 зійшлися з тими,що виводить PE TOOLS,тож я все зробила правильно.Також я більше дізналась про шифрування XOR і тепер вмію шифрувати дані за допомогою ключа і маю більш освідомлене ставлення до безпеки моїх даних.На скріншотах які я вище додала наглядно видно, що без шифрування відразу видно пароль,а у програмі с шифруванням він вже у зашифрованому вигляді.Ще я продовжую покращувати свої навички програмування на масм32.В загальному висновку,хочу сказати,що ця лабораторна була дуже корисною для мене,і я дізналась для себе дуже багато нового.