**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»

Виконав: Перевірив:

Студент групи ІМ-22 доц. Павлов В. Г.

Тимофеєв Даниіл Костянтинович

номер в списку групи: 23

Київ 2024

**Мета роботи:**

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

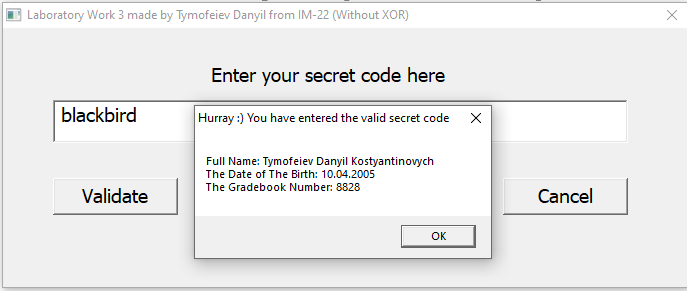
**Порядок виконання роботи:**

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.
5. Отриманий виконавчий файл дослідити за допомогою програми HEX-редактора HIEW32 (<https://uk.wikipedia.org/wiki/Hiew>) або WinHex (http://rainbowsky.ru/system/winhex/ - trial версия\*)[2].
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.
7. 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).
8. У останньому стовпчику таблиці розшифрувати виписані значення полів заголовка файлу. Таблицю привести в звіті по лабораторній роботі.
9. Провести дослідження того ж файлу за допомогою меню "PE Editor" безкоштовної програми PE Tools (http://soft.mydiv.net/win/download-PE-Tools.html\*). Все 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 побудовано 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.
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.
13. Зробити висновки по лабораторній роботі.

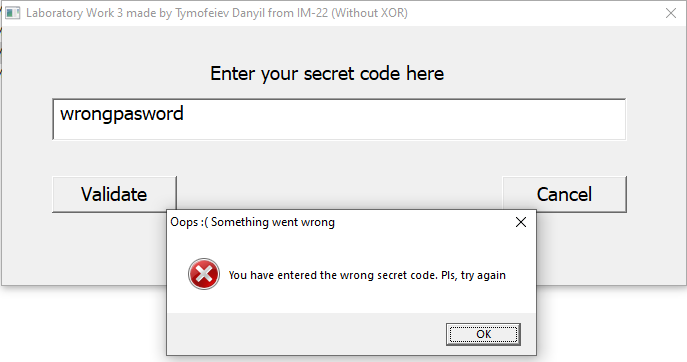
**Пункти 2-4.** Перевірити роботу програми шляхом введення як правильного, так і невірного паролів. Скріншоти програми (без шифрування XOR та із шифруванням пароля з XOR) :

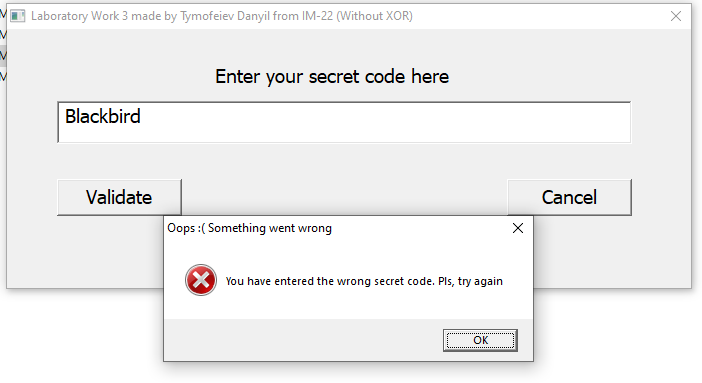
Пароль в обох програмах - blackbird

* Програма без шифрування XOR

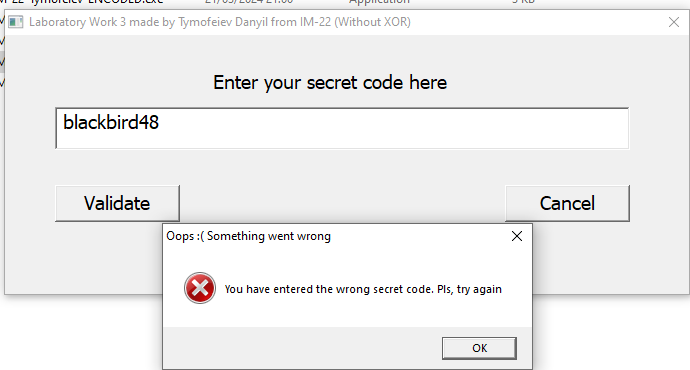
Введено правильний пароль (вивід інформації)

Введено неправильний пароль (повідомлення про помилку)

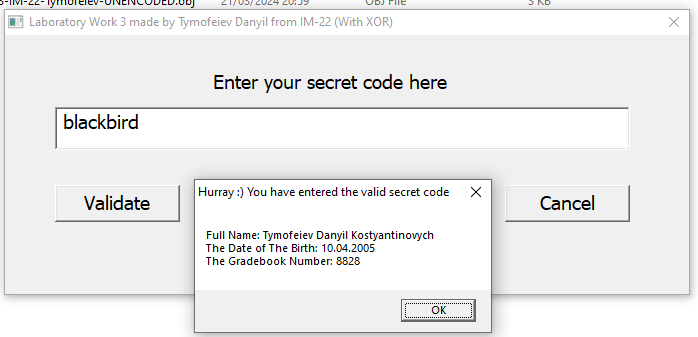


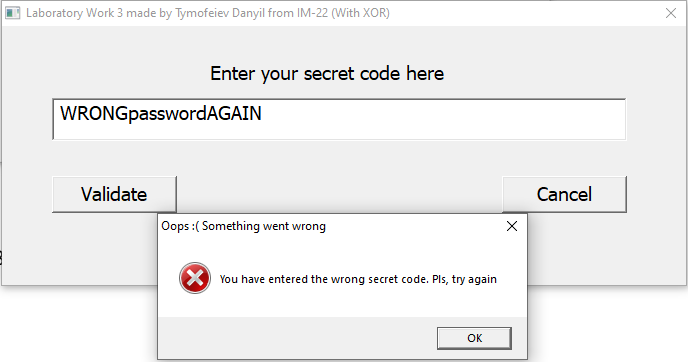
Введено неправильний пароль (перший символ у верхньому регістрі)

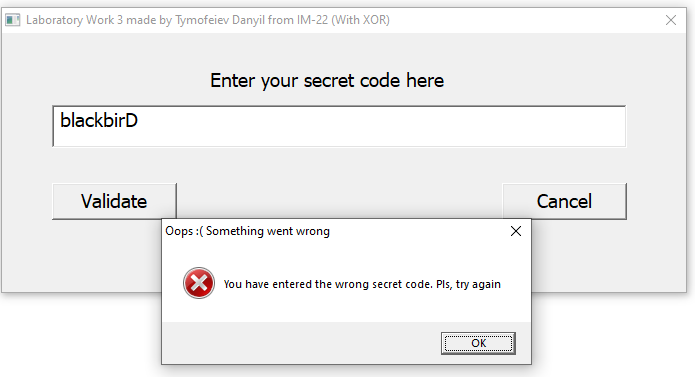
Введено неправильний пароль (додано зайві символи до кінця пароля)

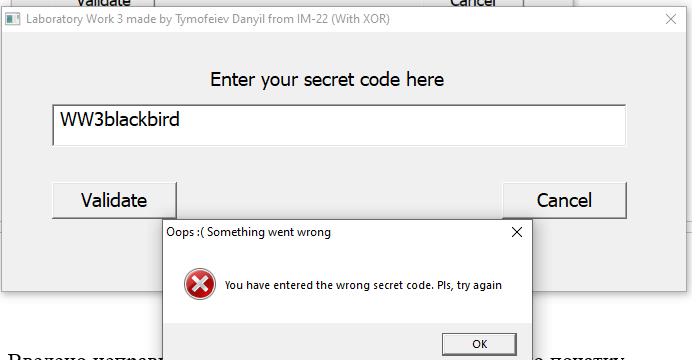


* Програма із шифруванням пароля з **XOR**:

Введено правильний пароль 

Введено неправильний пароль (повідомлення про помилку)

Введено неправильний пароль (останній символ у верхньому регістрі)

Введено неправильний пароль (додано зайві символи до початку пароля)

**Пункти 5-6.** Засобами WinHex**.** 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.

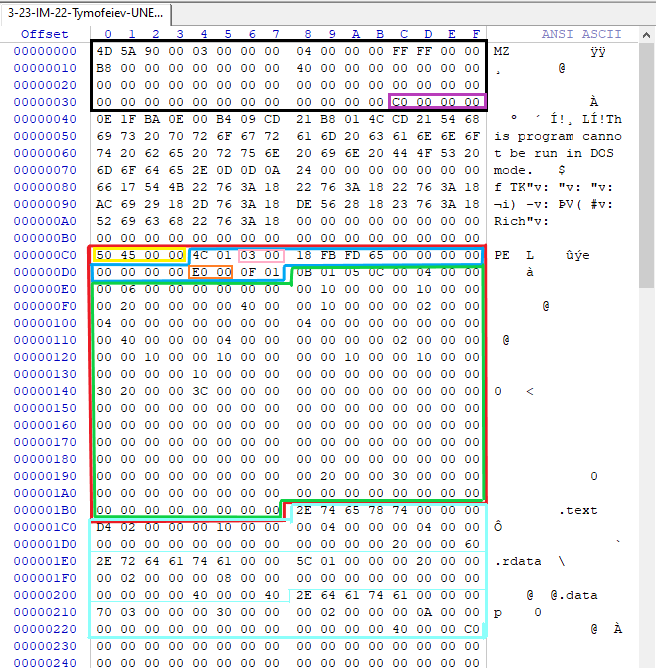
Використовуючи кольоровий олівець, введімо такі кольори для різних складових PE-файлу :

* DOS\_HEADER - **чорний** прямокутник
* Посилання на початок PE\_HEADER - фіолетовий прямокутник
* PE\_HEADER - червоний прямокутник
* PE Signature - жовтий прямокутник
* Заголовок PE-файлу - голубий прямокутник
* Optional Header - зелений прямокутник
* Розмір Optional Header - помаранчевий прямокутник
* Кількість Section Headers - рожевий прямокутник
* Section Headers - бірюзовий прямокутник

**DOS\_HEADER** розпочинає файл. **DOS\_HEADER** можна розпізнати за сигнатурою, тобто за 4 байтами 4D 5A (MZ у ASCII представленні). Розмір **DOS\_HEADER** - це 64 байти. Ба більше, **DOS\_HEADER** містить 4-байтовий зсув на початок PE\_HEADER (фіолетовий прямокутник) , який знаходиться за зсувом **0x3C.** Тож 4-байтовий зсув на початок PE\_HEADER (фіолетовий прямокутник) сягає у зворотному порядку **00 00 00 С0**, або 192 байти в десятковому представленні.

Далі перейдімо до PE\_HEADER (червоний прямокутник), який можна впізнати за 4-байтовою PE Signature (жовтий прямокутник), тобто за “PE\0\0”, + звичайно за посиланням з **DOS\_HEADER.** Далі перейдімо до заголовка PE-файлу (голубий прямокутник) з розміром 20 байтів. Optional Header (зелений прямокутник) його розмір можна визначити в заголовку PE-файлу в полі **SizeOfOptionalHeader**, зчитуємо розмір необов'язкового заголовку у зворотному порядку (помаранчевий прямокутник) - **00 E0,** тобто 224 байти.

Для визначення початку секції Section Headers (бірюзовий прямокутник) треба провести відлік байтів за кінцем Optional Header. Для визначення кількості секцій повернімося до заголовка PE-файлу (голубий прямокутник), де можна знайти параметр **NumberOfSections**, що займає 2 байти (рожевий прямокутник),тобто у зворотному порядку - **00 03**, або просто три секції, тобто спочатку йде **.text**, далі йде **.rdata** і закінчується **.data**. Секція **.text** містить програмний код; секція **.rdata** містить каталог відкладки; секція **.data** містить ініціалізовані дані. На малюнку розділено Section Headers (бірюзовий прямокутник) бірюзовими тонкими лініями на три відповідні секції, кожна з яких має розмір по 40 байтів, тобто по 2.5 рядків на кожну секцію.



**Пункти 7-8.** 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). У останньому стовпчику таблиці розшифрувати виписані значення полів заголовка файлу. Таблицю привести в звіті по лабораторній роботі.

Розділ 3.3.1 : **Machine Types**

| Константа | Значення | Опис |
| --- | --- | --- |
| Machine Type | 01 4С | Intel 386 або новішої версії та сумісні процесори |

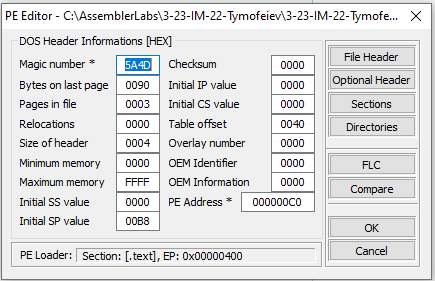
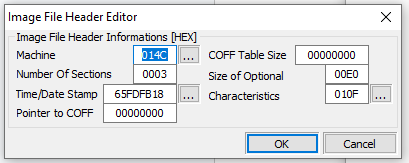
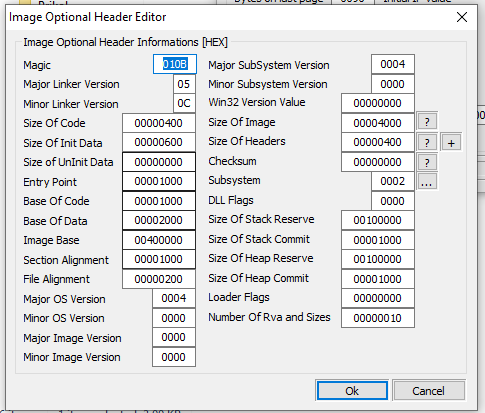
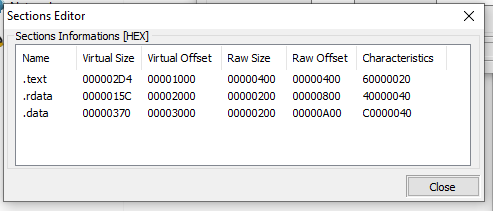
Розділ 3.4.1 : **Optional Header Standard Fields (Image Only)**

| Поля (Field) | Значення (Value) | Опис (Description) |
| --- | --- | --- |
| Magic | 01 0B | Беззнакове ціле число, що ідентифікує стан файлу. Найпоширенішим числом є 0413 вісімкове (0x10B), ідентифікуючи його як звичайний виконуваний файл |
| MajorLinkerVersion | 05 | Номер основної версії компонувальника |
| MinorLinkerVersion | 0C | Номер другорядної версії компонувальника |
| SizeOfCode | 00 00 04 00 | Розмір розділу коду або сума всіх розділів коду, якщо їх декілька |
| SizeOfInitializedData | 00 00 06 00 | Розмір ініціалізованого розділу даних або сума всіх таких розділів, якщо є декілька розділів даних |
| SizeOfUninitializedData | 00 00 00 00 | Розмір розділу неініціалізованих даних (BSS), або сума всіх таких розділів, якщо вони є кілька розділів BSS |
| AddressOfEntryPoint | 00 00 10 00 | Адреса точки входу відносно зображення під час завантаження виконуваного файлу пам'ять |
| BaseOfCode | 00 00 10 00 | Адреса, відносно основи зображення, початку розділу коду, коли завантажується в пам'ять |
| BaseOfData | 00 00 20 00 | Адреса, відносно бази зображення, початку розділу даних, коли завантажується в пам'ять |

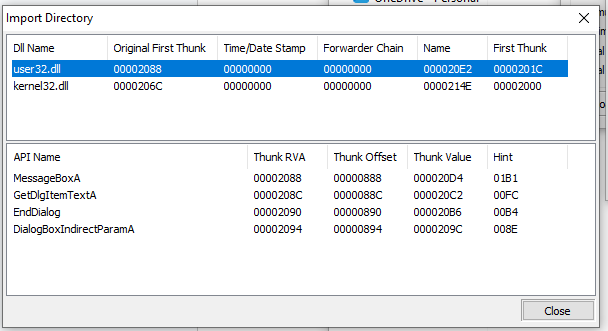
Розділ 4 : Section Table (Section Headers). Обрав секцію **.data**

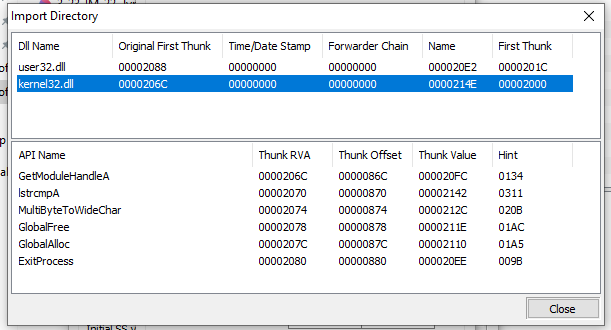
| Поля (Field) | Значення (Value) | Опис (Description) |
| --- | --- | --- |
| Name | .data\0\0\0 | 8-байтовий рядок ASCII із доповненням нульовими значеннями. Немає кінцевого нуля, якщо рядок містить рівно вісім символів. |
| VirtualSize | 00 00 03 70 | Загальний розмір розділу при завантаженні в пам'ять |
| VirtualAddress | 00 00 30 00 | Для виконуваних зображень це адреса першого байта розділу, коли він завантажується в пам'ять, відносно бази образу |
| SizeOfRawData | 00 00 02 00 | Розмір розділу (об'єктного файлу) або розмір ініціалізовані дані на диску (файли зображень) |
| PointerToRawData | 00 00 0A 00 | Покажчик файлу на першу сторінку розділу у файлі COFF |
| PointerToRelocations | 00 00 00 00 | Покажчик файлу на початок записів переміщення для розділу. Встановіть 0 для виконуваних зображень або якщо немає переміщень |
| PointerToLinenumbes | 00 00 00 00 | Покажчик файлу на початок записів номерів рядків для розділу. Встановіть значення 0, якщо немає номерів рядків COFF |
| NumberOfRelocatios | 00 00 | Кількість записів про переміщення для розділу. Встановіть 0 для виконуваних зображень |
| NumberOfLinenumbs | 00 00 | Кількість рядків-номер записів для розділу |
| Characteristics | C0 00 00 40 | Прапори, що описують характеристики розділу |

**Пункт 9.** Провести дослідження того ж файлу за допомогою меню "PE Editor" безкоштовної програми PE Tools (http://soft.mydiv.net/win/download-PE-Tools.html\*). Все скріншоти вікон програми з даними, відповідними раніше побудованій таблиці, привести в звіті по лабораторній роботі.

* Інформація про DOS Header : 
* Інформація про Image File Header : 
* Інформація про Image Optional Header File : 
* Інформація про Sections : 

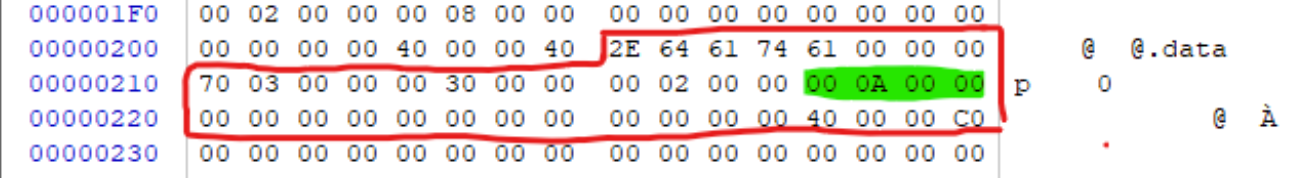
Тож, як можна побачити, дані з таблиць та дані з PE Tools повністю збігаються.

**Пункт 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.

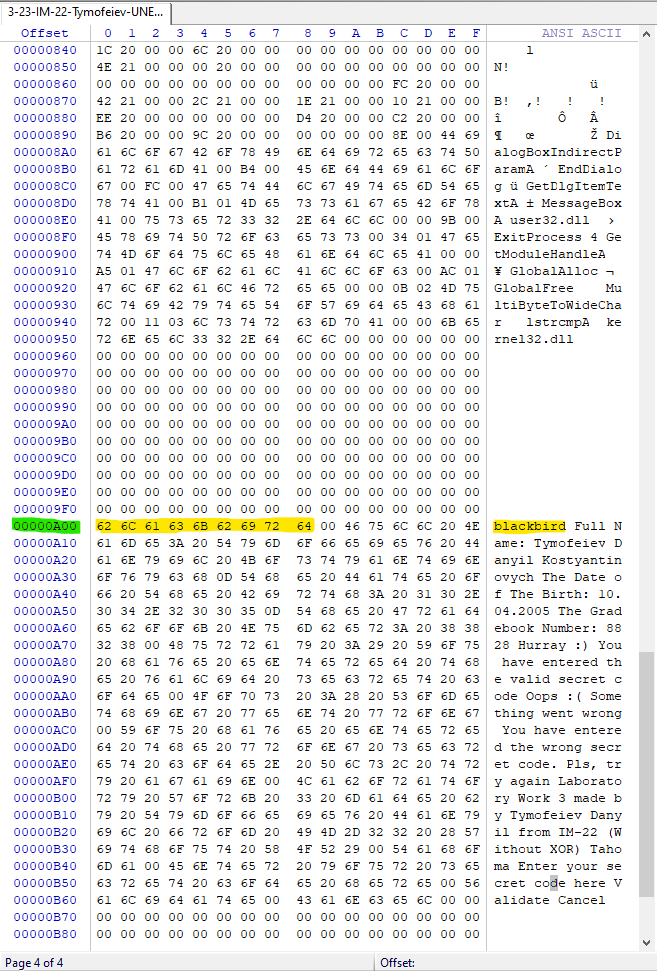


Тож, на малюнках вище можна побачити, що файл імпортує дві бібліотеки, а саме **user32.dll** і **kernel32.dll.** Підключена бібліотека **user32.dll** використовує такі функції: MessageBoxA, GetDlgItemTextA, EndDialog, DialogBoxIndirectParamA. Підключена бібліотека **kernel32.dll** використовує такі функції: GetModuleHandleA, lstrcmpA, MultiByteToWideChar, GlobalFree, GlobalAlloc, ExitProcess.

**Пункт 11**. Знайти в тексті файлу по зсуву, узятому з побудовано 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.

На наведеному фрагменті скріншоту представлена секція **.data** (червоним) з виділеним зсувом **00 00 0A 00** (зелений; див. таблицю).

Тепер за зсувом **00 00 0A 00** можна знайти текст оригіналу пароля та переконатися, що його можна легко виявити за допомогою HEX-редактора. У скріншоті нижче зеленим кольором виділений зсув **00 00 0A 00**, а жовтим виділений пароль у HEX - коді та в ASCII представленні.



**Пункт 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.

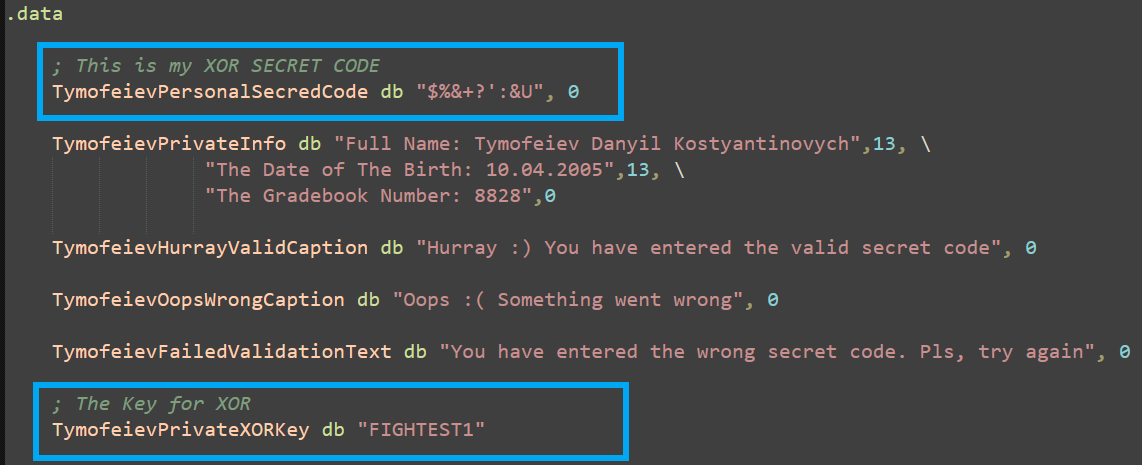
Виконаємо шифрування пароля “blackbird” за допомогою **XOR** з використанням ключа “FIGHTEST1”**:**

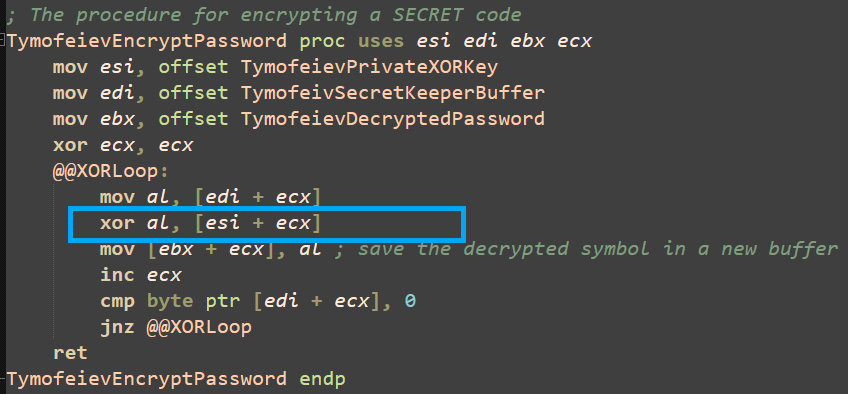
1) Переведемо “blackbird” з ASCII в двійкову систему : 01100010 01101100 01100001 01100011 01101011 01100010 01101001 01110010 01100100

2) Переведемо “FIGHTEST1” з ASCII в двійкову систему : 01000110 01001001 01000111 01001000 01010100 01000101 01010011 01010100 00110001

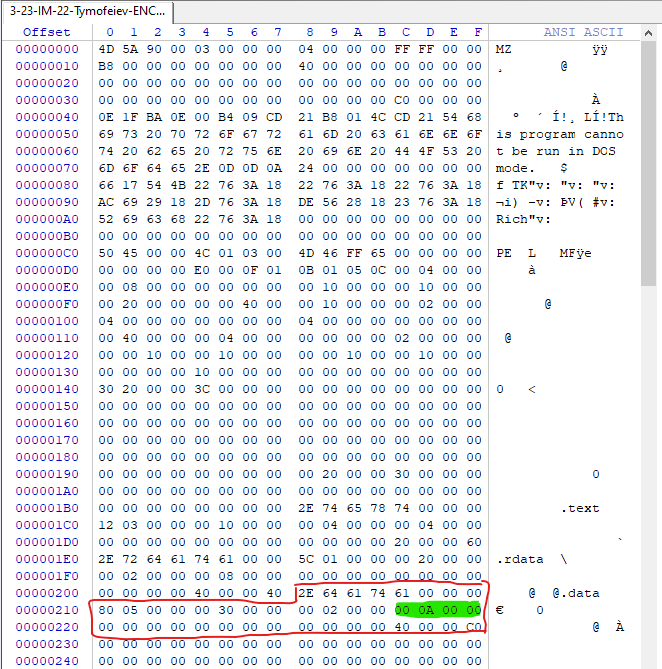
3) Зробимо XOR : 00100100 00100101 00100110 00101011 00111111 00100111 00111010 00100110 01010101

4) Переведемо в ASCII : “$%&+?':&U”

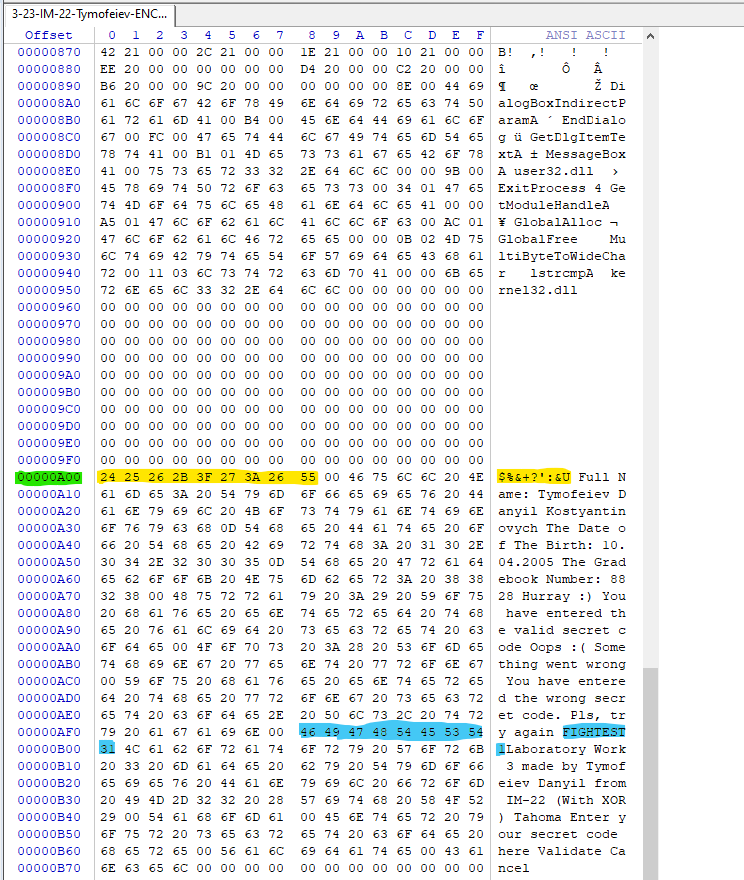
Тож, “$%&+?':&U” - зашифрований пароль. У коді не буде тепер пароля в прямому вигляді, а буде зашифрований + ключ для шифрування

Ще у програмі роблю XOR над введеним паролем та ключем

Знову знайдемо зсув **00 00 0A 00** (зелений; див. таблицю) у файлі із зашифрованим паролем у секції **.data** (червоний)



Тепер за зсувом **00 00 0A 00** переконаймося, що пароль у файлі лежить вже не у відкритому виді, а в зашифрованому. Зеленим позначений зсув **00 00 0A 00,** жовтим зашифрований пароль у **Hex** та **Text** режимах. Також синім було виділено ключ для XOR в **Hex** та **Text** режимах.



**Пункт 13. Висновки**

Виконуючи лабораторну роботу, я ознайомився зі специфікацією **COFF** (Common Object File Format); використав прийоми дослідження структури файлів PE-формату. Отримав навички роботи з HEX-редакторами на прикладі **WinHex**. Навчився знаходити заголовок **DOS**, визначати його розмір; аналізувати заголовок **PE** з його складовими (сигнатура, заголовок **PE-файлу** та необов’язковий заголовок). Ще було досліджено таблицю імпортів з бібліотеками та функціями, які використовує файл. Ба більше, під час аналізу **Section Headers** було виявлено зсув на реальні дані в програмі, у нашому випадку це пароль.

Ще однією важливою складовою було створення таблиці за стандартом “Microsoft Portable Executable and Common Object File Format Specification”, куди заносилися параметри **.EXE** файлу. Табличні дані було порівняно з даними, отриманими програмою **PE Tools.** Помітив, що дані з двох джерел збігаються, тож можна сказати, що ті чи інші параметри було знайдено правильно.

Також при виконанні роботи було розроблено дві програми на **MASM32** для отримання персональних даних за валідним паролем без шифрування та із шифрування за допомогою функції **XOR**. Порівнюючи ці дві програми, можна сказати, що програма із шифрування за допомогою функції **XOR** є більш надійною, оскільки ми зберігаємо пароль не в явному виді, тобто в коді нашої програми, а вже в зашифрованому виді. Цей підхід є більш надійним, у порівнянні зі зберіганням пароля без шифрування. Хоча треба відмітити, що використання функції **XOR** не є дуже надійним підходом, з точки зору систем безпеки сучасних ПЗ, бо можна зробити розшифрування зашифрованого пароля за допомогою ключа, який також лежить у відкритому доступі, у коді нашої програми.