**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 ІІ курсу ФІОТ доц. Павлов В. Г.

групи ІМ-22

Сачко Максим Євгенійович

номер у списку групи (варіант): 20

Київ 2024

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

**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. Запустити вбудований редактор qeditor.exe, а у ньому відкрити консольне вікно.**

Правильний пароль – ZeusOlympus

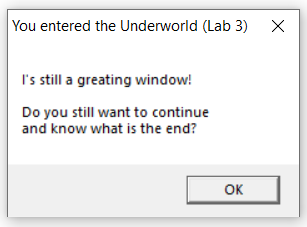
**Скріншоти виконання програми:**

Незашифрована програма (UNENCRYPTED):

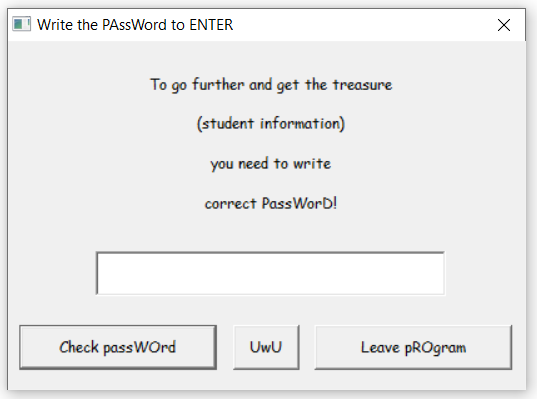
Вікно при запуску програми (прев’ю)



Друге вікно після запуску програми (це теж прев’ю)

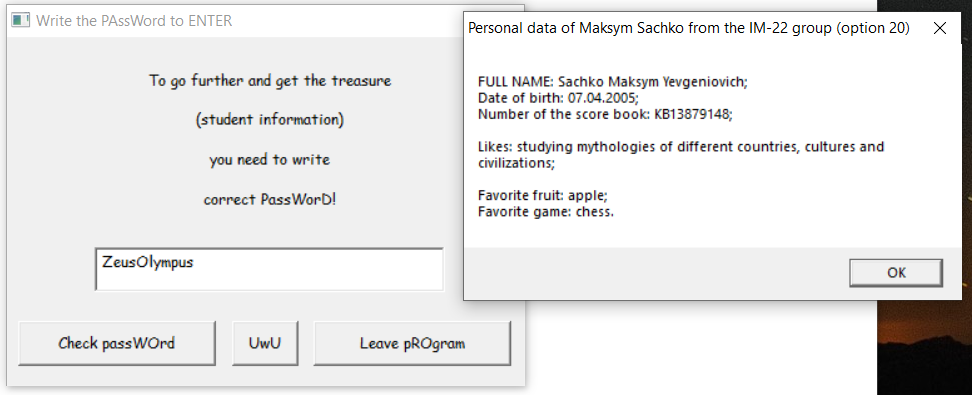


Вікно для введення паролю

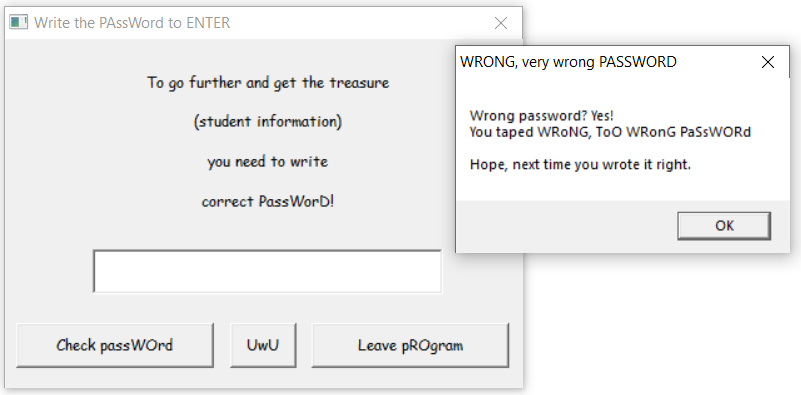


Введення паролю:

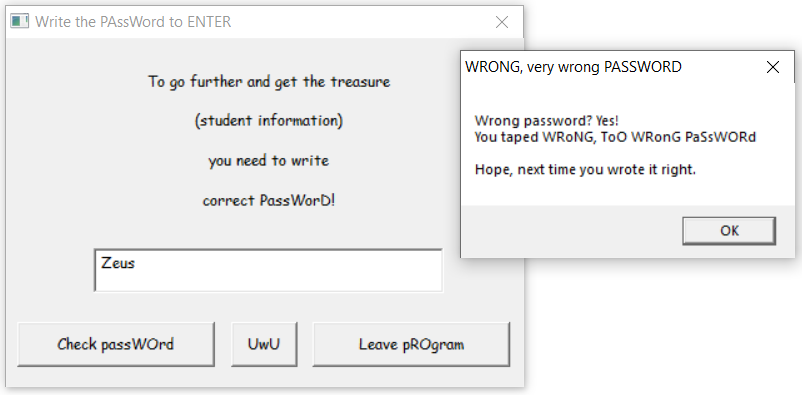
1. Введення правильного паролю



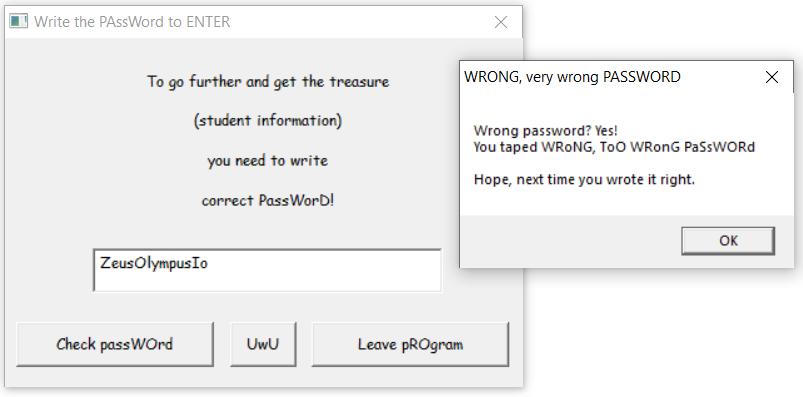
2. Введення неправильного паролю (пустий рядок)



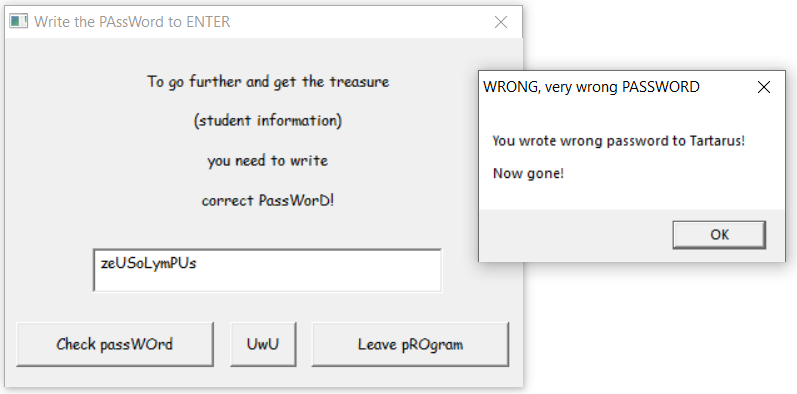
3. Введення неправильного паролю (неповний пароль)



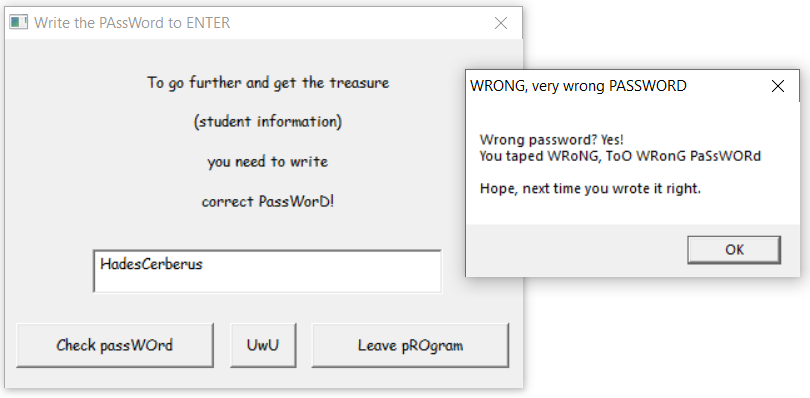
4. Введення неправильного паролю (пароль з надлишковою кількістю літер)



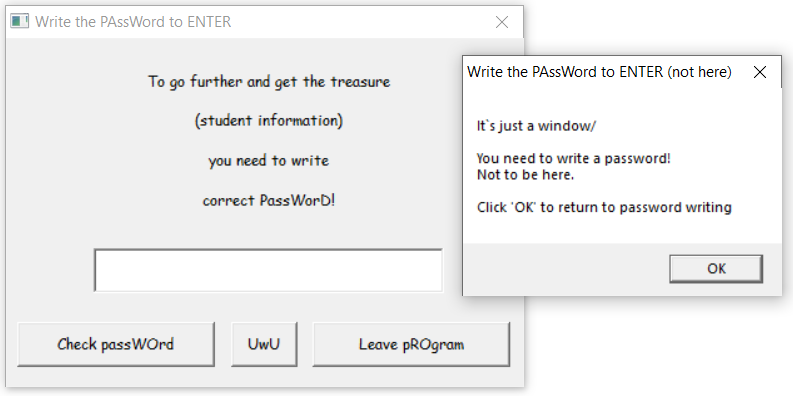
5. Введення неправильного паролю (пароль з іншим регістром деяких символів)



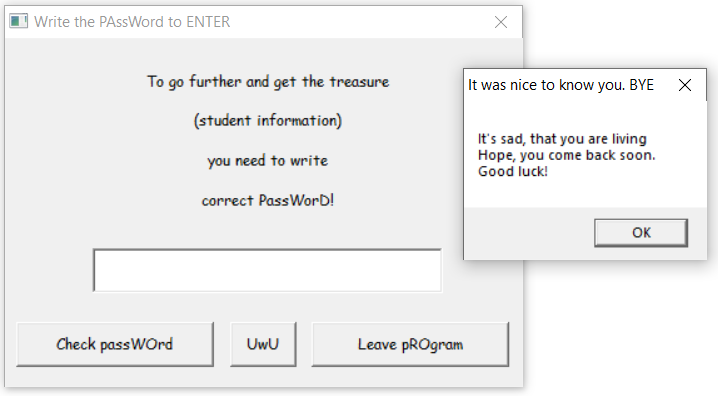
6. Введення неправильного паролю (зовсім інший пароль)



Вікно, яке з’являється після натискання кнопки “UwU” (**після закриття цього вікна вікно з введенням паролю залишиться відкритим**)

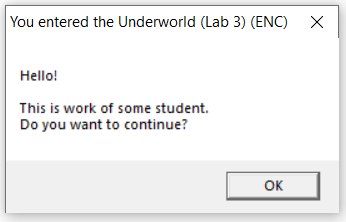


Вікно, яке з’являється після виходу з програми за допомогою кнопки “ Leave pROgram”

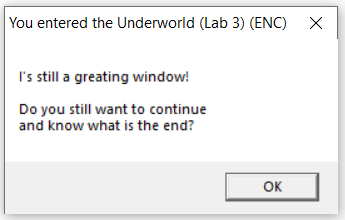


Зашифрована програма (ENCRYPTED):

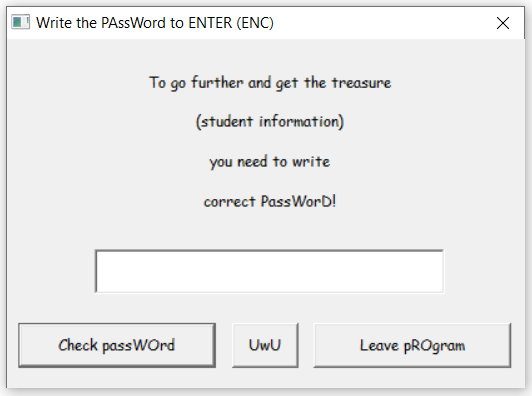
Вікно при запуску програми (прев’ю)



Друге вікно після запуску програми (це теж прев’ю)

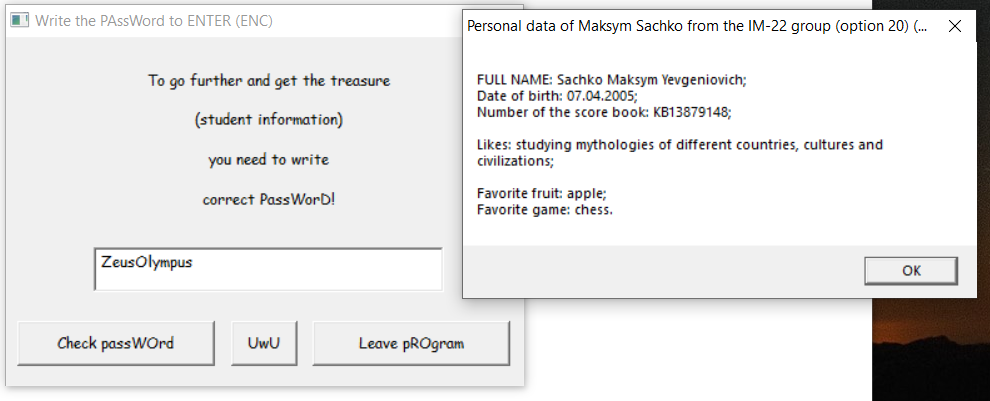


Вікно для введення паролю

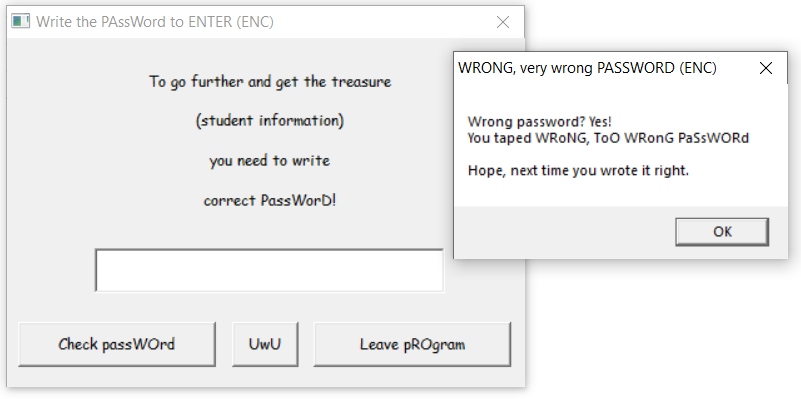


Введення паролю:

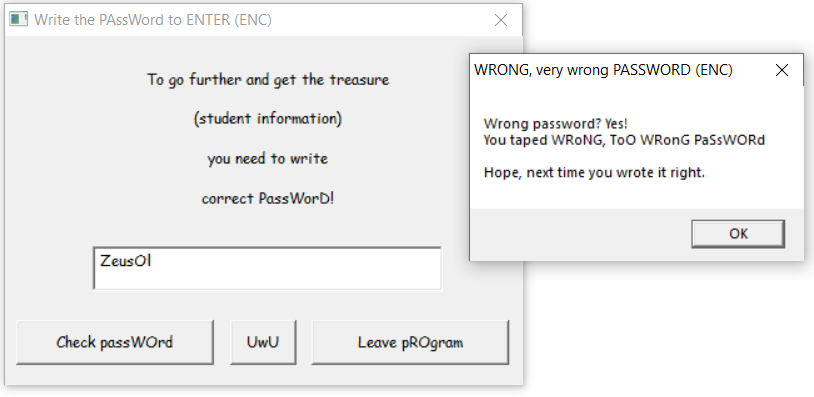
1. Введення правильного паролю



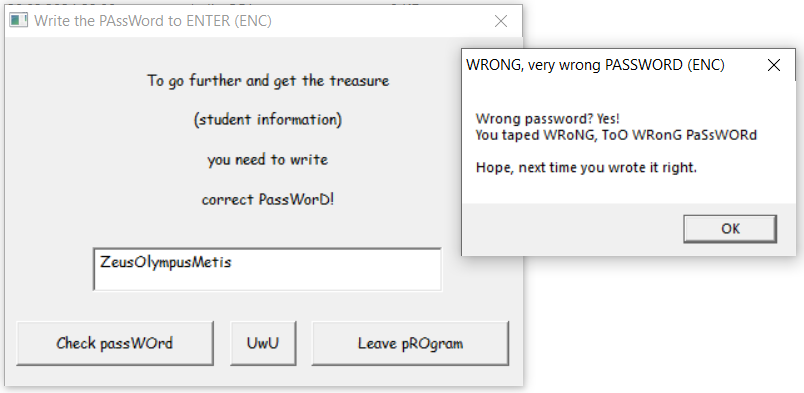
2. Введення неправильного паролю (пустий рядок)



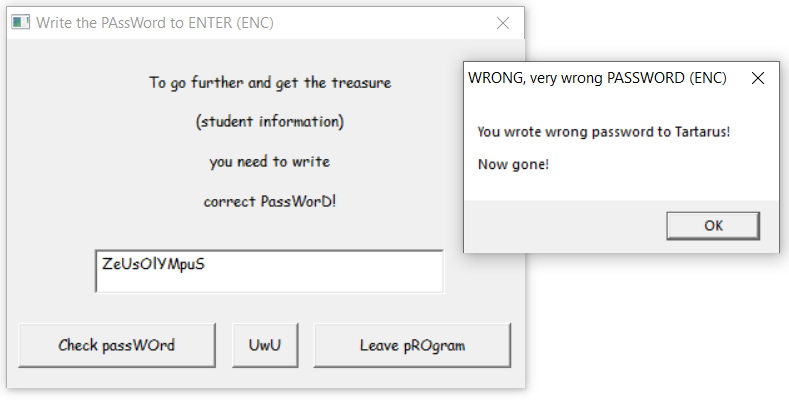
3. Введення неправильного паролю (неповний пароль)



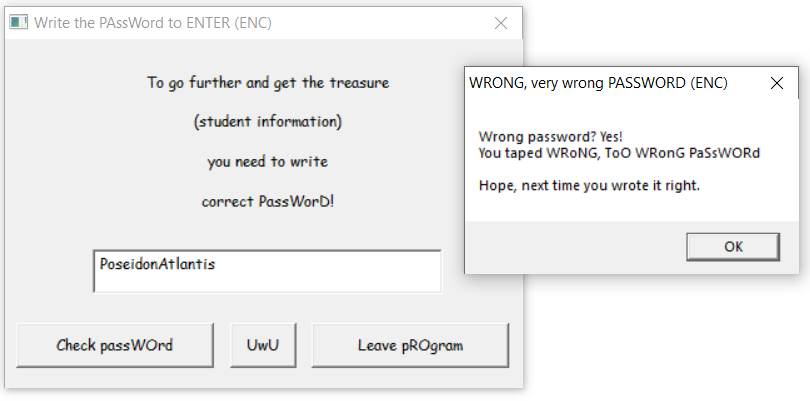
4. Введення неправильного паролю (пароль з надлишковою кількістю літер)



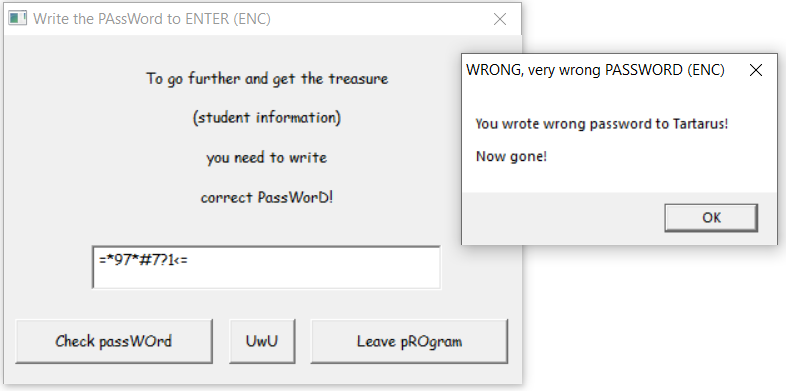
5. Введення неправильного паролю (пароль з іншим регістром деяких символів)



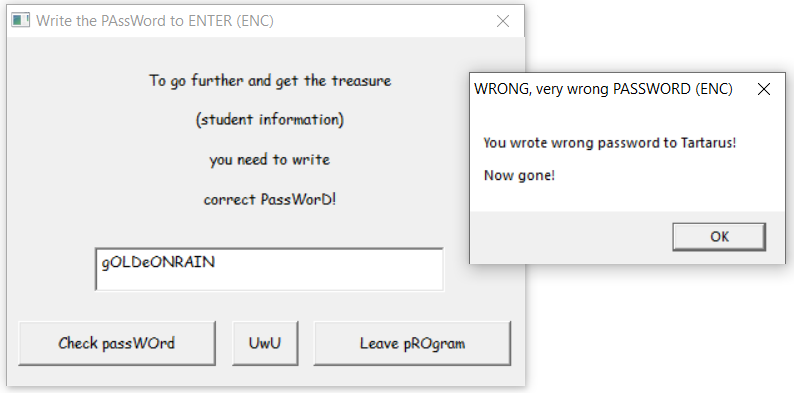
6. Введення неправильного паролю (зовсім інший пароль)



7. Введення неправильного паролю (зашифрований пароль) (**те, як пароль був зашифрований, буде наведено у звіті нижче в 12 пункті**)



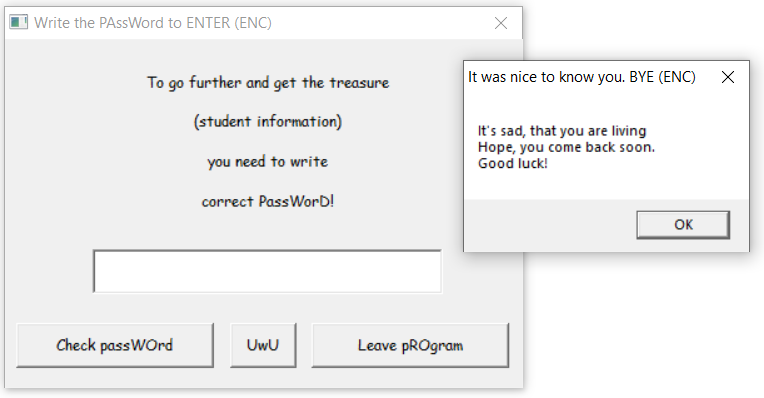
8. Введення неправильного паролю (ключ)



Вікно, яке з’являється після натискання кнопки “UwU” (**після закриття цього вікна вікно з введенням паролю залишиться відкритим**)



Вікно, яке з’являється після виходу з програми за допомогою кнопки “ Leave pROgram”

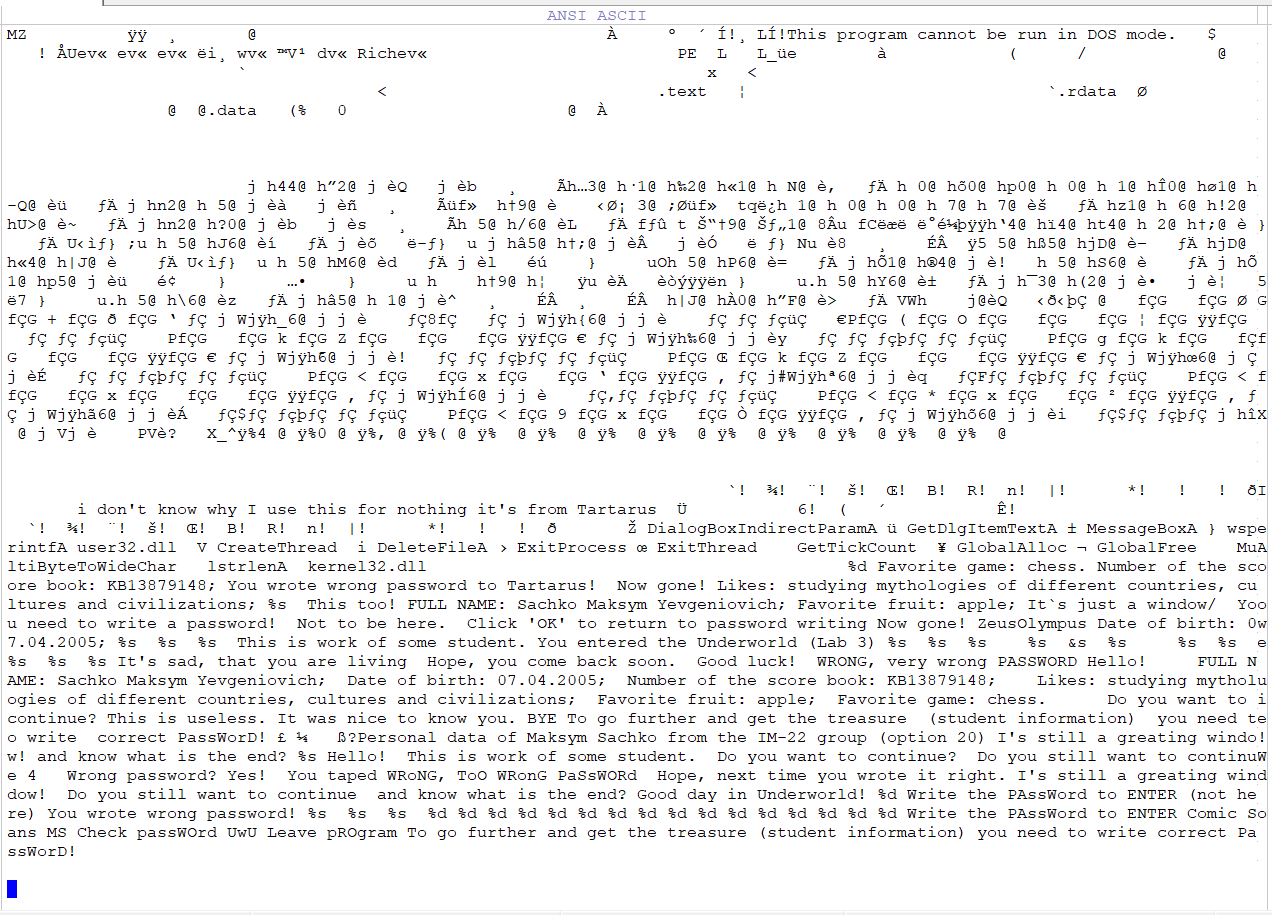


**5. Отриманий виконавчий файл дослідити за допомогою програми HEX-редактора HIEW32 (**[**https://uk.wikipedia.org/wiki/Hiew**](https://uk.wikipedia.org/wiki/Hiew)**) або WinHex (**[**http://rainbowsky.ru/system/winhex/**](http://rainbowsky.ru/system/winhex/) **– trial версия\*)[2].**

**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.**

Дослідження файлу в WinHex

Формат «Text»



Формат «Hex»

На скріншоті обведено області MS-DOS заголовка (DOS\_HEADER, **зеленим** кольором), PE заголовка (PE\_HEADER, **оранжевим** кольором) і таблиці секцій (SECTION\_HEADERS, **рожевим** кольором).

DOS\_HEADER розпочинається на початку файла, його розмір сягає 64 байт. В його останніх 4 байтах – вказівник на початок PE\_HEADER (обведено **коричневим** кольором). Враховуючи, що байти записано в зворотному порядку, зсув для початку PE\_HEADER дорівнює 00 00 00 C0, тобто 192 байти (початок PE\_HEADER обведено **сірим** кольором).

PE\_HEADER розпочинається з PE Signature (обведено **блакитним** кольором), розміром у 4 байти, в якому записано байтову сигнатуру “PE\0\0”. Після PE Signature іде заголовок PE-файлу (обведено **чорним** кольором), який має розмір у 20 байтів. Після нього йде Optional Header (позначено **фіолетовим** кольором). Його розмір визначено в заголовку PE-файлу в SizeofOptionalHeader (обведено **салатовим** кольором), він займає 2 байти, і в нашому випадку, звертаючи увагу на те, що байти записано в зворотньому порядку, його значення сягає 00 E0h, тобто 224 байти.

Одразу після Optional Header іде Section Headers (обведено **рожевим** кольором). Кількість секцій у ньому записано в заголовку PE-файлу в NumberofSections (ці дані займають 2 байти, на скріншоті обведено **жовтим** кольором), звертаючи увагу на те, що байти записано в зворотньому порядку, це значення дорівнює 00 03, тобто всього три секції (.text, .rdata, .data) (секції розділені **рожевими** лініями), розмір кожної з яких сталий і сягає 40 байтів. Отже, розмір Section Headers: 3 \* 40 = 120 байт.



**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).**

Розділ 3.3.1: Machine Types (перші 2 байти в заголовку PE-файлу)

|  |  |  |
| --- | --- | --- |
| Поле | Значення | Опис поля |
| Machine type | 01 4C | Тип машини (CPU): Intel 386 або пізніша, та інші схожі процесори |

Розділ 3.4.1: Optional Header Standard Fields (Image Only)

|  |  |  |
| --- | --- | --- |
| Поле | Значення | Опис поля |
| Magic | 01 0B | Беззнакове ціле число (unsigned int), що ідентифікує стан файлу.  0x10B – ідентифікатор звичайного виконуваного файлу (.exe). |
| MajorLinkerVersion | 05 | Номер основної версії компонувальника |
| MinorLinkerVersion | 0C | Номер другорядної версії компонувальника.  0С = 1210 |
| SizeOfCode | 00 00 08 00 | Розмір секції коду або сума всіх розділів коду, якщо їх є декілька |
| SizeOfInitializedData | 00 00 28 00 | Розмір ініціалізованого розділу даних або сума всіх таких розділів, якщо їх декілька розділи даних. |
| SizeOfUninitializedData | 00 00 00 00 | Розмір розділу неініціалізованих даних (BSS), або сума всіх таких розділів, якщо є декілька розділів BSS |
| AddressOfEntryPoint | 00 00 13 2F | Адреса вхідної точки в програму, початкова адреса програми |
| BaseOfCode | 00 00 10 00 | Адреса початку секції коду при завантаженні в пам’ять |
| BaseOfData | 00 00 20 00 | Адреса початку секції змінних при завантаженні в пам’ять |

Розділ 4: Section Table (Section Headers)

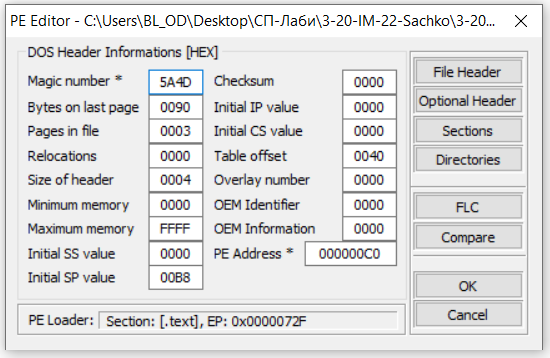
Як приклад, по порядку розібрано третю секцію .data

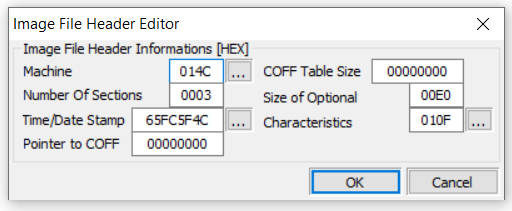
|  |  |  |
| --- | --- | --- |
| Поле | Значення | Опис поля |
| Name | .data\0\0\0 | 8-байтовий рядок ASCII із доповненням нульовими значеннями |
| VirtualSize | 00 00 25 28 | Загальний розмір розділу при завантаженні в пам'ять |
| VirtualAddress | 00 00 30 00 | Адреса першого байту розділу |
| SizeOfRawData | 00 00 08 00 | Розмір розділу |
| PointerToRawData | 00 00 0E 00 | Вказівник на першу сторінку розділу в COFF файл |
| PointerToRelocations | 00 00 00 00 | Вказівник файлу на початок записів переміщення для розділу (0 для .exe) |
| PointerToLinenumbers | 00 00 00 00 | Вказівник файлу на початок записів номерів рядків для розділу |
| NumberOfRelocations | 00 00 | Кількість релокаційних входжень в розділ (0 для .ехе) |
| NumberOfLinenumbers | 00 00 | Кількість номерних записів розділу |
| Characteristic | C0 00 00 40 | Прапори, що описують характеристики розділу |

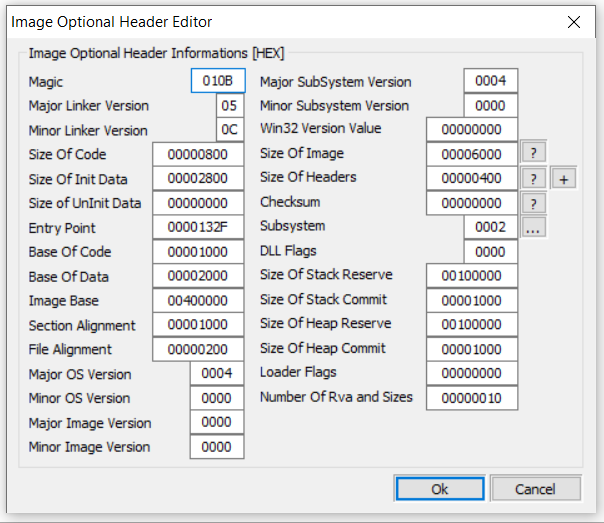
**8. У останньому стовпчику таблиці розшифрувати виписані значення полів заголовка файлу. Таблицю привести в звіті по лабораторній роботі.**

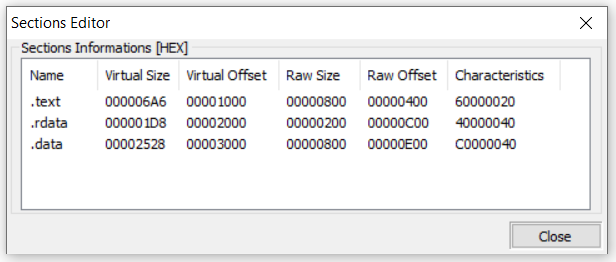
Зроблено у пункті 7

**9. Провести дослідження того ж файлу за допомогою меню "PE Editor" безкоштовної програми PE Tools (**[**http://soft.mydiv.net/win/download-PE-Tools.html**](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.**



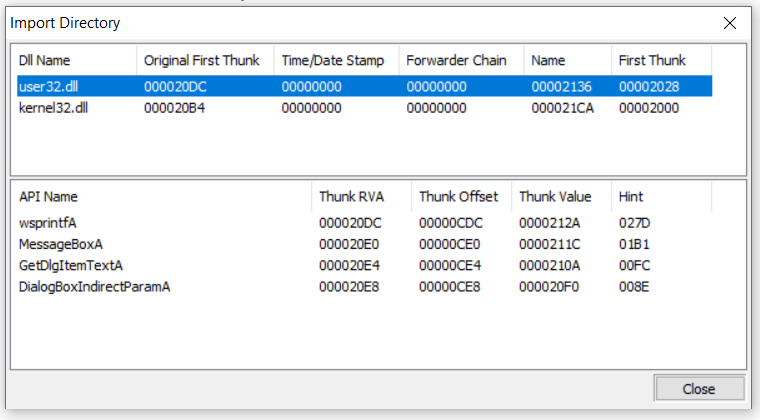


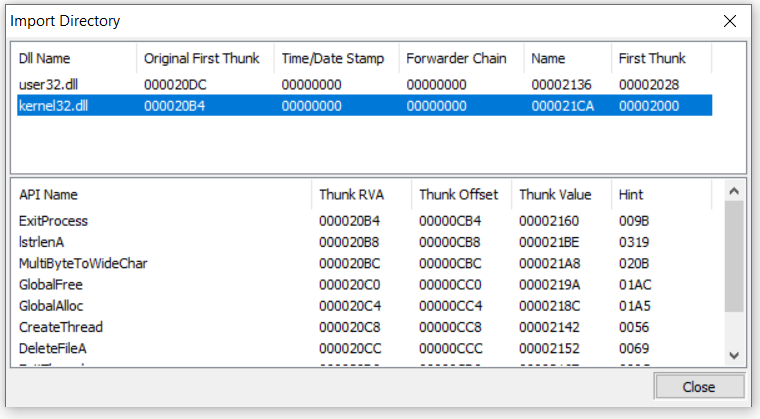


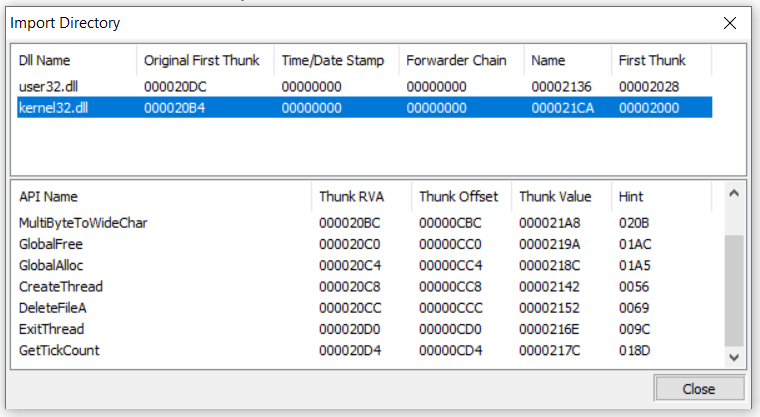


**10. Дослідити таблицю імпорту (Import Directory) даного файлу і визначити, які саме функції використовуються з бібліотек, що підключаються. Скріншоти вікон Import Directory з функціями, що імпортуються, з кожного бібліотечного файлу привести в звіті по лабораторній роботі.**

В програмі імпортується дві бібліотеки: user32.dll та kernel32.dll, нижче зображено список функцій, що були використані за допомогою цих бібліотек, що використовується в програмі, знайдений за допомогою PE Tools.

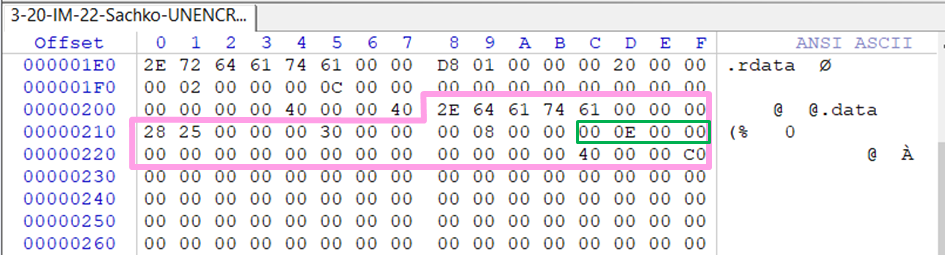




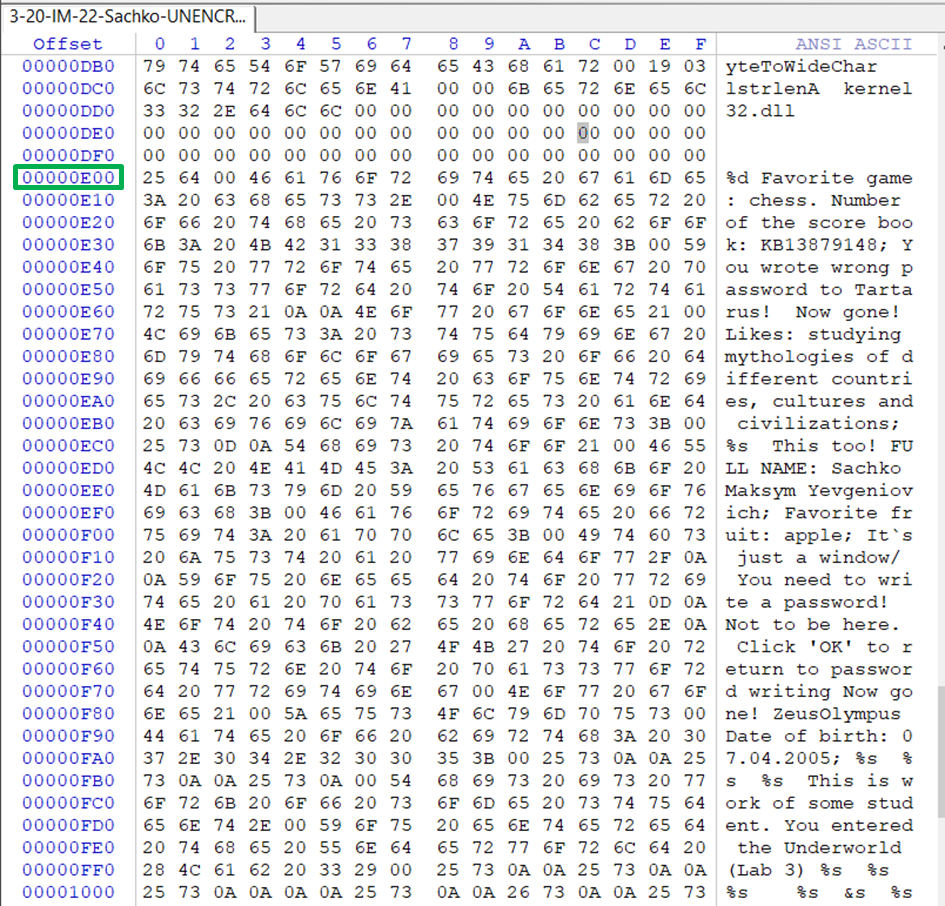


**11. Знайти в тексті файлу по зсуву, узятому з побудованої таблиці, секцію з даними і переконатися, що текст оригінала пароля, що міститься в тексті програми, може бути легко виявлений за допомогою HEX-редактора. Привести скріншот цього фрагмента програми у вигляді HEX - коду в звіті по лабораторній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

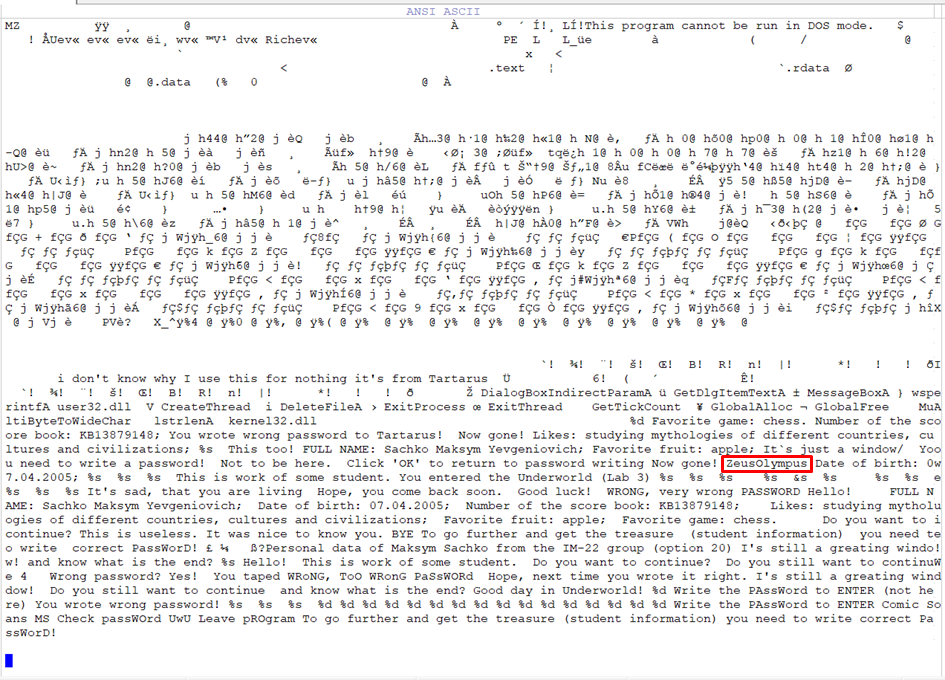
Зсув розташовано у секції .data, у полі PointerToRawData (знайдено у пункті 5) з таблиці Section Table (Section Headers), починається за зсувом 20 байт відносно початку секції. Розмір PointerToRawData складає 4 байти. Враховуючи, що байти записано в зворотному порядку, зсув для початку секції з даними дорівнює 00 00 0E 00, тобто 3584 байти.



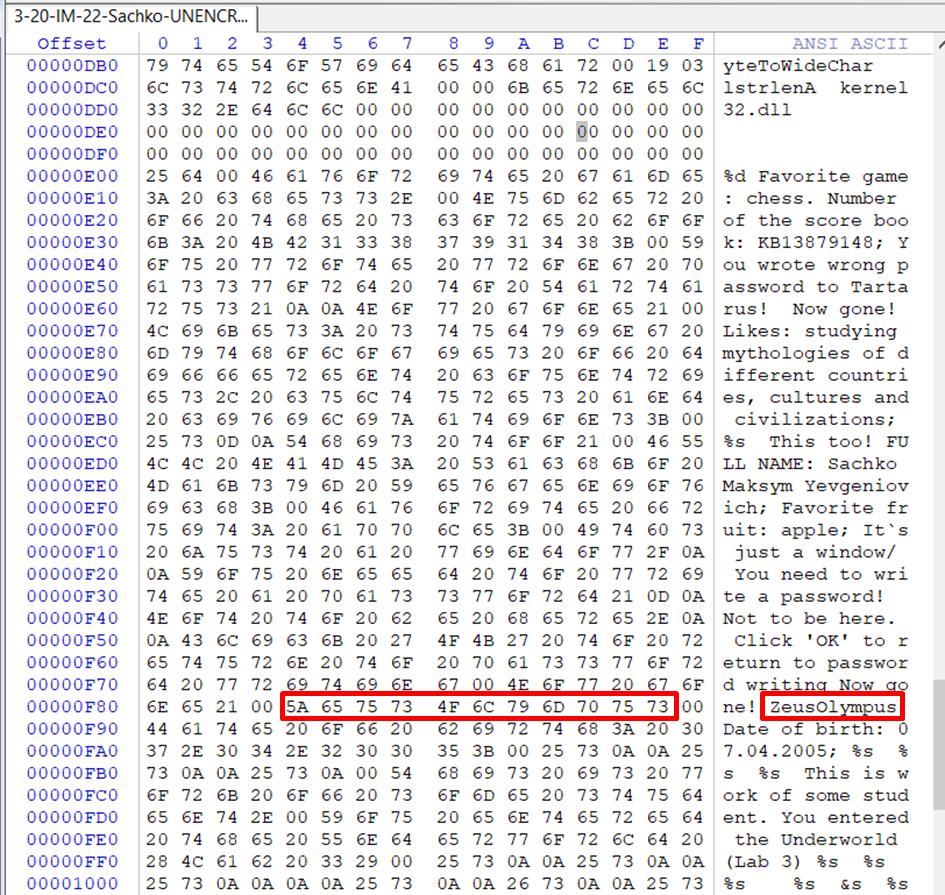
На цьому скріншоті видно, що секція з даними починається саме з цього зсуву:



Знайдений пароль у секції з даними (режим «Text»):



Знайдений пароль у секції з даними (режим «Hex»):



Як видно зі скріншота, за допомогою HEX-редактора у файлі програми можна досить легко знайти оригінал пароля, що є не дуже добре в досвіді реальної розробки.

**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.**

Схема шифрування:

У циклі шифруються усі символи пароля за допомогою базового метода шифрування XOR з ключем “gOLDeONRAIN”.

Ключ: gOLDeONRAIN

Пароль: ZeusOlympus

Переведення символів у двійкову систему координат використовуючи ASCII таблицю:

gOLDeONRAIN:

g = 10310 = 0110 01112

O = 7910 = 0100 11112

L = 7610 = 0100 11002

D = 6810 = 0100 01002

e = 10110 = 0110 01012

O = 7910 = 0100 11112

N = 7810 = 0100 11102

R = 8210 = 0101 00102

A = 6510 = 0100 00012

I = 7310 = 0100 10012

N = 7810 = 0100 11102

ZeusOlympus:

Z = 9010 = 0101 10102

e = 10110 = 0110 01012

u = 11710 = 0111 01012

s = 11510 = 0111 00112

O = 7910 = 0100 11112

l = 10810 = 0110 11002

y = 12110 = 0111 10012

m = 10910 = 0110 11012

p = 11210 = 0111 00002

u = 11710 = 0111 01012

s = 11510 = 0111 00112

Шифрування:

Z XOR g = 0101 10102 XOR 0110 01112 = 0011 11012 = 6110 = **=**

e XOR O = 0110 01012 XOR 0100 11112 = 0010 10102 = 4210 = **\***

u XOR L = 0111 01012 XOR 0100 11002 = 0011 10012 = 5710 = **9**

s XOR D = 0111 00112 XOR 0100 01002 = 0011 01112 = 5510 = **7**

O XOR e = 0100 11112 XOR 0110 01012 = 0010 10102 = 4210 = **\***

l XOR O = 0110 11002 XOR 0100 11112 = 0010 00112 = 3510 = **#**

y XOR N = 0111 10012 XOR 0100 11102 = 0011 01112 = 5510 = **7**

m XOR R = 0110 11012 XOR 0101 00102 = 0011 11112 = 6310 = **?**

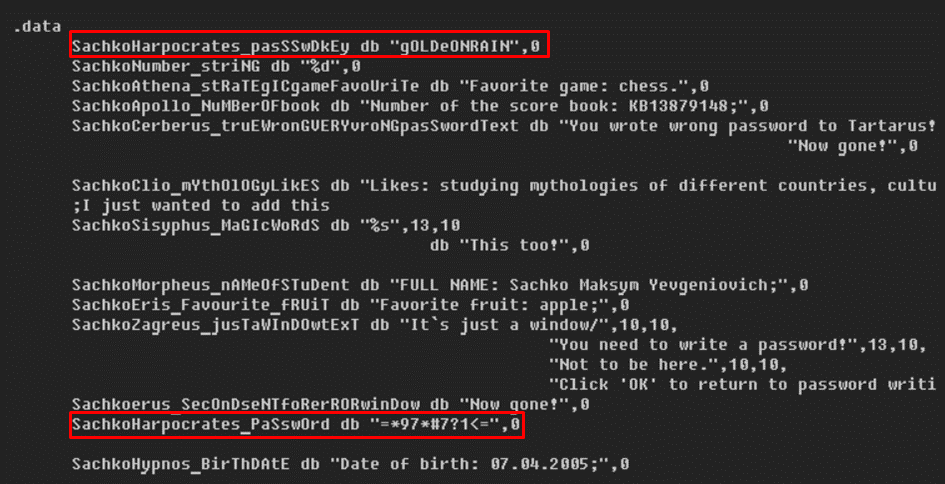
p XOR A = 0111 00002 XOR 0100 00012 = 0011 00012 = 4910 = **1**

u XOR I = 0111 01012 XOR 0100 10012 = 0011 11002 = 6010 = **<**

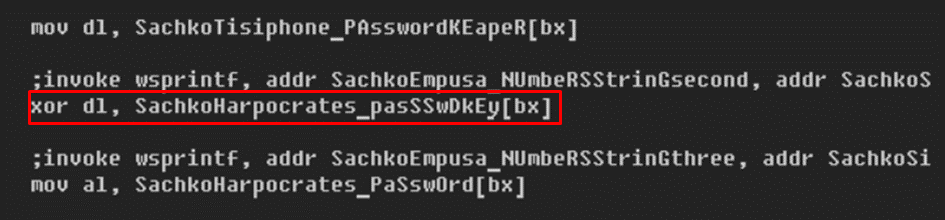
s XOR N = 0111 00112 XOR 0100 11102 = 0011 11012 = 6110 = **=**

Отже, зашифрований пароль – **=\*97\*#7?1<=**

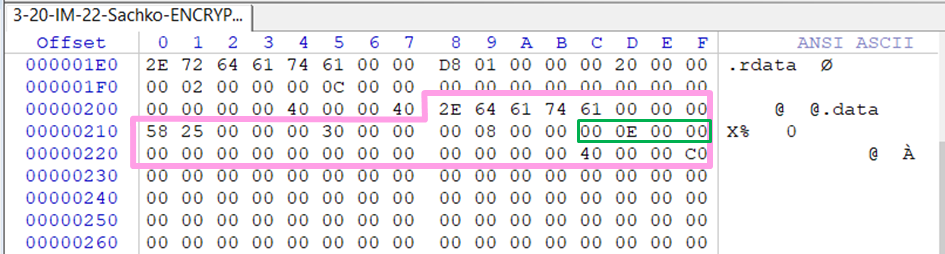
На скріншоті показано фрагмент коду, де я визначаю зашифрований пароль та ключ для шифрування вводу користувача для порівняння.

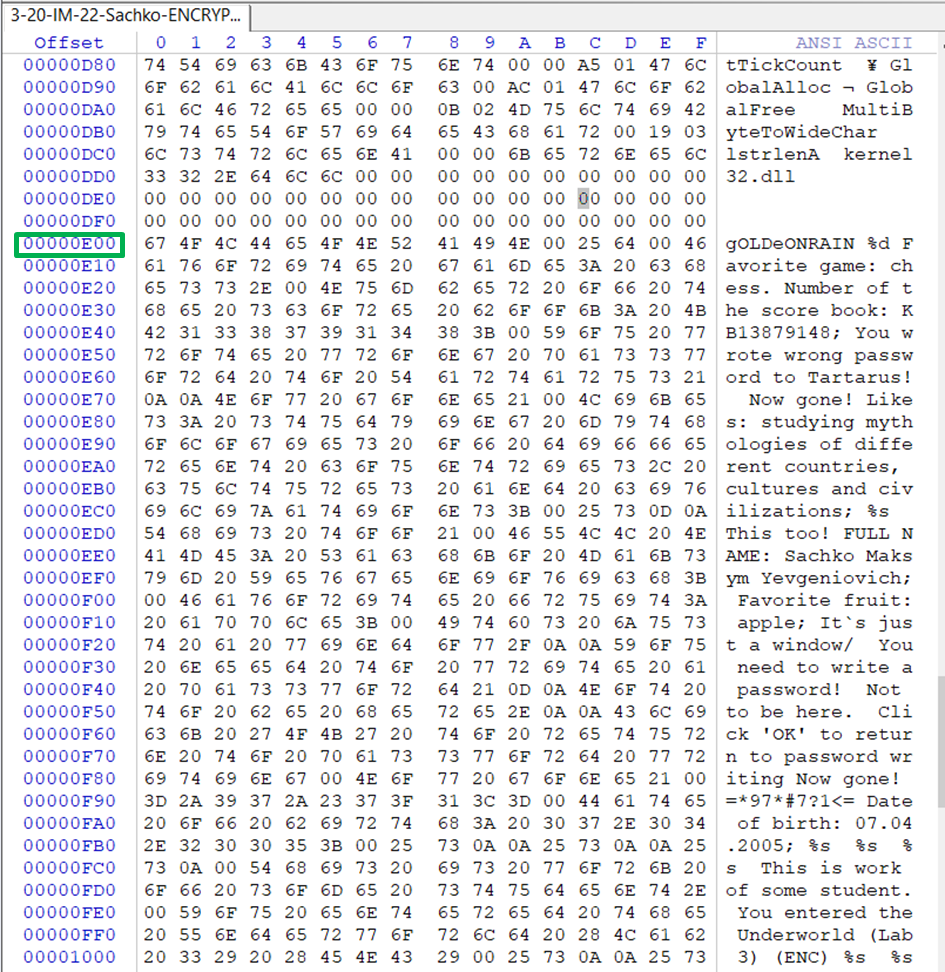


Далі у коді програми я додав рядок, який робить операцію xor з кожним байтом введеного користувачем пароля:

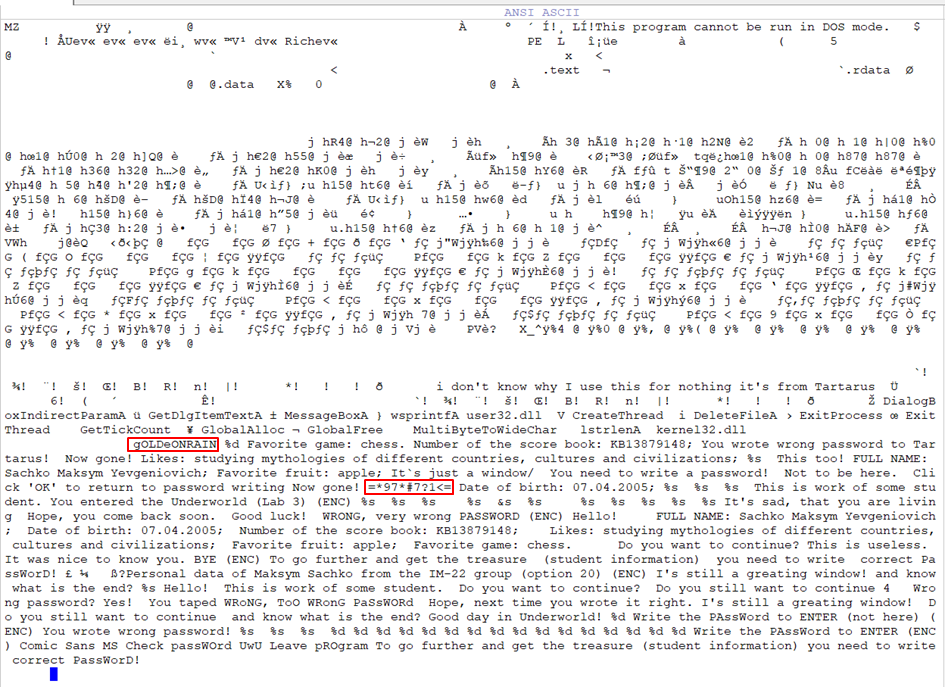


Як і в програмі без шифрування, секція з даними у програмі з шифруванням також починається зі зсуву 00 00 0E 00:

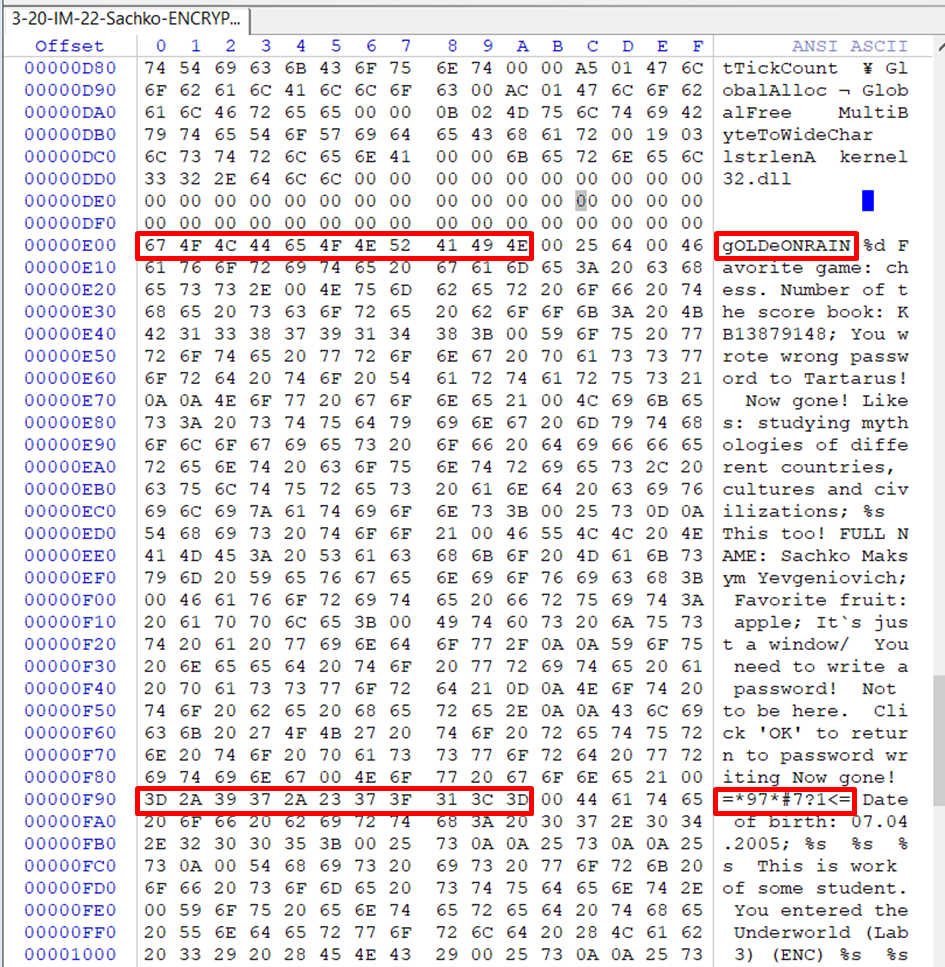




Знайдений зашифрований пароль та ключ у секції з даними (режим «Text»):



Знайдений зашифрований пароль та ключ у секції з даними (режим «Hex»):



Тепер якщо подивитися вміст файлу програми з шифруванням, побачимо, що в ній немає явно записаного пароля, а є лише зашифрований пароль та ключ, які знаходяться на деякій відстані один від одного, завдяки чому визначити справжній пароль буде набагато складніше:

**13. Зробити висновки по лабораторній роботі.**

**Висновок:** Під час виконання даної лабораторної роботи я отримав нові знання щодо часткового аналізу exe-файлів, зокрема:

* Встановлення положення (початок файла) та розміру (64 байти) DOS-заголовка.
* Виявлення початку PE-заголовка (зсув для нього вказано в 4 останніх байтах DOS-заголовка), його компонентів та їх розмірів, якщо вони фіксовані, або визначення місцезнаходження цих розмірів, якщо вони змінні. Також вивчення розміру (40 байт \* кількість секцій) та розташування (одразу після PE-заголовка) Section Headers.
* Встановлення початку секцій у пам'яті для використання в певних програмах (у моєму випадку WinHex та PE Tools), наприклад, для пошуку місцезнаходження пароля у програмі, яка вимагає введення пароля для отримання даних. У результаті порівняння даних, знайдених мною власноруч у файлі програми за допомогою WinHex з даними, які я отримав з PE Tools, я роблю висновок, що я правильно визначив місцезнаходження та значення полів заголовків файла, оскільки досліджені власноруч дані співпадають із даними, отриманими з PE Tools.
* Крім того, ознайомлення з методом XOR-шифрування даних. Незважаючи на те, що цей метод не є надійним, оскільки зашифрований пароль та ключ можуть бути знайдені у файлі, і пароль можна розшифрувати (хоча це займе певний час) і пароль не буде прямо вказаний. Тож у результаті порівняння двох програм (без застосування шифрування та з ним) я впевнився, що застосування шифрування є більш безпечною практикою, оскільки без нього пароль просто можна знайти у виконуваному файлі програми, а з шифруванням справжній пароль дізнатися буде складніше, оскільки явно у файлі зберігатися буде лише зашифрований пароль. Однак, крім зашифрованого пароля також у файлі програми можна знайти й ключ для розшифрування (оскільки шифрування методом XOR є операцією, яка працює як у бік шифрування, так і розшифрування), тому в реальній практиці цей метод не набагато надійніший за відсутність шифрування. Натомість для шифрування на практиці існують більш надійні методи, наприклад, із використанням закритого ключа.