**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 №4

з дисципліни «Системне програмування» на тему

“Макровизначення і макроси в MASM32”

Виконав: Перевірив:

Студент ІІ курсу ФІОТ доц. Павлов В. Г.

групи ІМ-12

Сутулов Нікіта Олегович

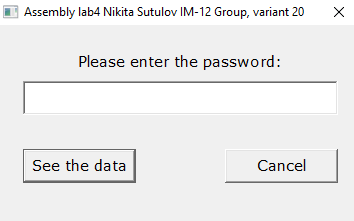
номер у списку групи (варіант): 20

номер залікової книжки: 1229

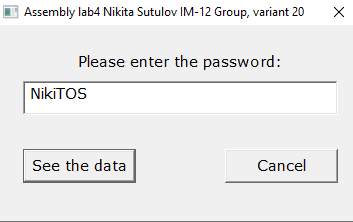
Київ 2023

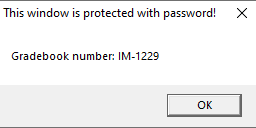
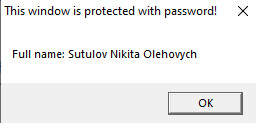
**Скріншоти виконання програми**

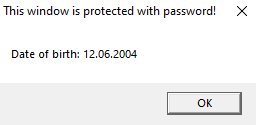
При запуску програми з’являється вікно, в яке треба ввести пароль для отримання подальших даних:



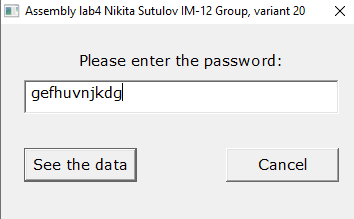
При введенні правильного пароля кожний фрагмент даних (ім’я, номер залікової книжки, дата народження) тепер виводиться в своєму вікні послідовно завдяки реалізації макроса 1, а не як у 3 лабораторній, коли всі дані показувалися в одному вікні:

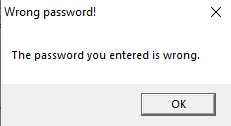






Якщо ввести неправильний пароль, то вікно з повідомленням про неправильний пароль буде також створено за допомогою макроса 1:





Відповідно до завдання лабораторної роботи, мною було визначено три макроси:

* **SutulovShowOutput** – виводить на екран повідомлення з даними в разі введення правильного пароля, або ж повідомлення про те, що пароль неправильний, приймає два аргументи: текст вікна в аргументі message та назва вікна в аргументі title;
* **SutulovEncrypt** – здійснює шифрування пароля;
* **SutulovCompare** – здійснює порівняння введеного пароля та зашифрованого.

**Шифрування**

У програмі ітеративно відбувається шифрування кожного байта пароля з відповідним байтом ключа за допомогою базового метода - XOR

Пароль: NikiTOS

Ключ: aBRAham

Переведення символів у двійкову систему використовуючи ASCII таблицю:

| N = 7810 = 010011102  i = 10510 = 011010012  k = 10710 = 011010112  i = 10510 = 011010012  T = 8410 = 010101002  O = 7910 = 010011112  S = 8310 = 010100112 | a = 9710 = 011000012  B = 6610 = 010000102  R = 8210 = 010100102  A = 6510 = 010000012  h = 10410 = 011010002  a = 9710 = 011000012  m = 10910 = 011011012 |
| --- | --- |

Процес шифрування:

N = 010011102 XOR a = 011000012 => 001011112 = 4710 = /

i = 011010012 XOR B = 010000102 => 001010112 = 4310 = +

k = 011010112 XOR R = 010100102 => 001110012 = 5710 = 9

i = 011010012 XOR A = 010000012 => 001010002 = 4010 = (

T = 010101002 XOR h = 011010002 => 001111002 = 6010 = <

O = 010011112 XOR a = 011000012 => 001011102 = 4610 = .

S = 010100112 XOR m = 011011012 => 001111102 = 6210 = >

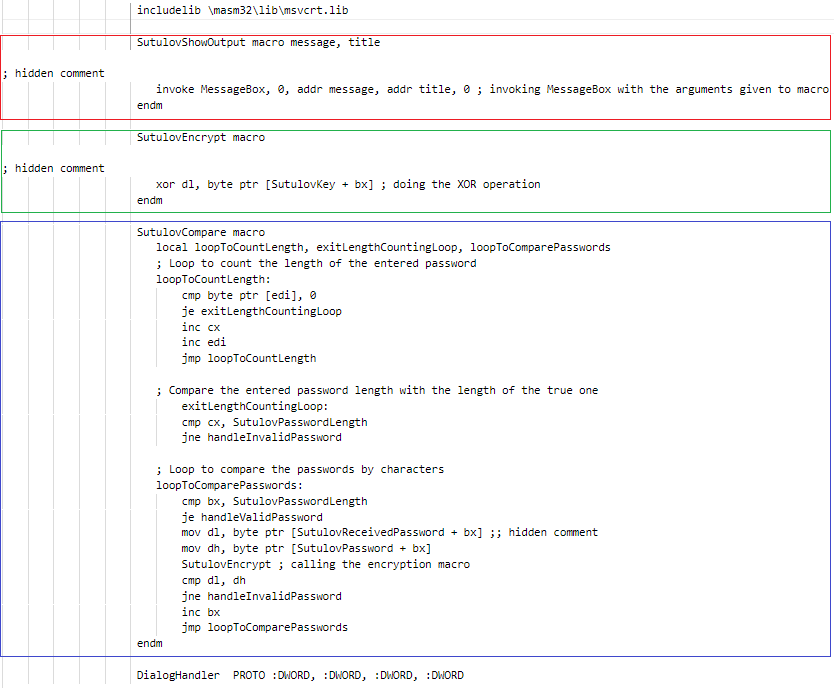
Отже, зашифрований пароль – /+9(<.>

**Дослідження лістингу**

Варіант програми з макросами, які написано разом з рештою коду

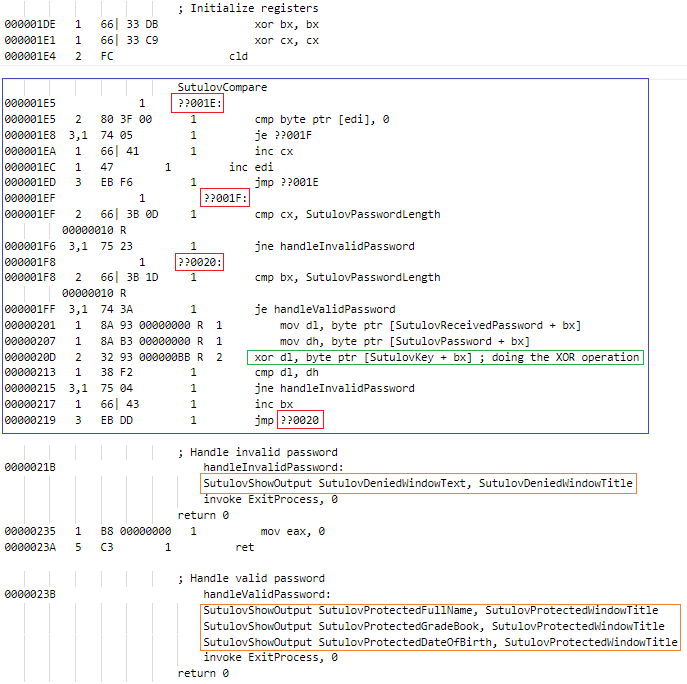
**Макровизначення**

Макровизначення мною було визначено після підключення бібліотек та перед оголошенням сигнатури функції, яка обробляє поведінку вікна. Видно, що ми можемо побачити як звичайні, так і приховані коментарі.



На скріншоті показано та обведено макровизначення для виведення вікна з повідомленням (червоним кольором), шифрування пароля (зеленим кольором) та порівняння паролів (синім кольором). Варто зауважити на цьому етапі, що у макросі для порівняння паролів використано локальні мітки, і пізніше буде розглянуто їхню поведінку при трансляції.

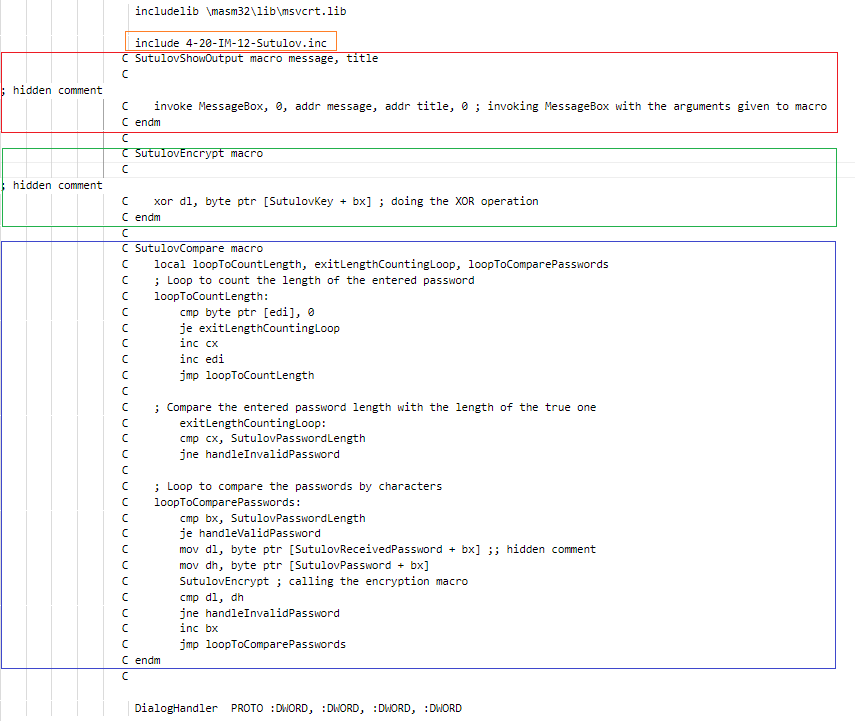
**Трансляції макросів**



На скріншоті вище показано трансляції для всіх трьох макросів: для першого макроса (виведення повідомлення у вікно) обведено оранжевим кольором, для другого (шифрування пароля) – зеленим, для третього (порівняння паролів) – синім. Так як другий макрос викликається всередині третього, то він транслюється у цей третій макрос порівняння, а тому його назви SutulovEncrypt не бачимо. Також примітно, що у третьому макросі замість назв локальних міток при трансляції з’явилися коди (їх обведено червоним кольором). Крім цього, варто зауважити, що після трансляції не видно приховані коментарі, натомість звичайні коментарі присутні. Насамкінець, для першого макроса (виведення повідомлення у вікно), можна побачити, що команди invoke, що містяться у цьому макросі, у лістингу при трансляції приховано, а так як це єдиний рядок, з якого складається даний макрос, то при трансляції ми бачимо тільки виклик цього макроса з передачею йому аргументів.

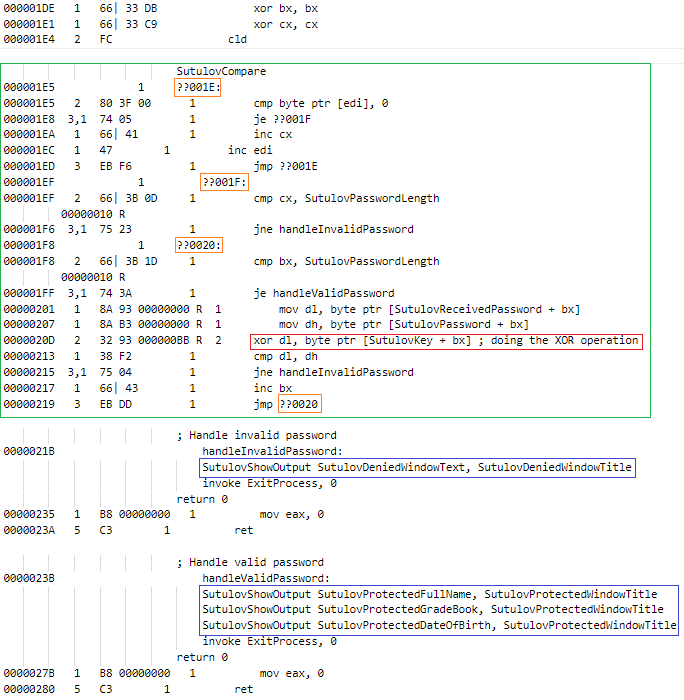
Тепер дослідимо варіант програми, у якому макроси записано в окремий файл, і їх імпортовано за допомогою команди include

**Макровизначення**



У лістингу другого варіанта програми макровизначення показуються одразу після команди include для їхнього імпорту (обведено оранжевим кольором). На скріншоті вище видно макроси для виведення повідомлення у вікно (обведено червоним кольором), для шифрування пароля (зеленим кольором) та порівняння паролів (синім кольором). Як і в першому варіанті програми, тут видно як звичайні, так і приховані коментарі. Також далі побачимо, що станеться з локальними мітками при трансляції.

**Трансляції макросів**



На скріншоті лістингу другого варіанту програми спостерігаємо загалом те саме, що і для першого варіанту. Знову ж таки, показано трансляції для всіх трьох макросів: для першого макроса (виведення повідомлення у вікно) обведено синім кольором, для другого (шифрування пароля) – червоним, для третього (порівняння паролів) – зеленим. Також, так як другий макрос викликається всередині третього, то він транслюється у цей третій макрос порівняння, а тому його назви SutulovEncrypt не бачимо. Крім цього, варто відмітити, що у третьому макросі замість назв локальних міток при трансляції з’явилися коди (їх обведено оранжевим кольором). Після трансляції, як і в першому варіанті програми, не видно приховані коментарі, натомість звичайні коментарі присутні. Насамкінець, для першого макроса (виведення повідомлення у вікно), можна побачити, що команди invoke, що містяться у цьому макросі, у лістингу при трансляції також приховано, і враховуючи, що це єдиний рядок, який входить до цього макроса, цілком зрозуміло, що при трансляції буде видно лише виклик самого макроса з передачею йому аргументів.

**Висновки**

У ході виконання даної лабораторної роботи я дізнався, що таке макроси в мові асемблера MASM32, і як їх використовувати, а також як їх можна різними шляхами додавати до своєї програми: або прописуючи у самому файлі програми разом з рештою коду, або ж прописати їх в окремому файлі, який включити до головного файлу за допомогою команди include. Також я познайомився з локальними мітками, назви яких при трансляції у лістингу замінюються на унікальні коди.

Порівнюючи два варіанти програми (з визначенням макросів в одному файлі разом з рештою коду та визначенням макросів у окремому файлі з подальшим його імпортом у головний файл), можу визначити деякі схожості та відмінності між ними.

*Що в них схожого:*

* у макровизначеннях в обох варіантах видно як звичайні, так і приховані коментарі;
* при трансляції в обох варіантах приховані коментарі більше не видно;
* назви локальних міток замінюються при трансляції на унікальні коди в обох варіантах програми;
* рядки з викликом функцій за допомогою команди invoke при трансляції в обох варіантах приховано – це якраз добре видно на прикладі першого макроса, який складається лише з виклику функції MessageBox за допомогою команди invoke, тому при трансляції залишається тільки рядок з викликом цього макроса із вказанням аргументів;
* у трансляціях макросів їхній код вставляється одразу під їхнім викликом.

*Що в них відмінного:*

* єдиною ключовою відмінністю є розташування макровизначень у файлах лістингу.

У результаті, можна стверджувати, що в обох варіантах трансляція однакова, єдина відмінність полягає в розташуванні макровизначень у розширеному лістингу.

Насамкінець можу сказати, що макроси в MASM32 - це невід'ємний інструмент для розробників, який значно спрощує створення блоків коду, які можна використовувати багаторазово. Вони дозволяють ефективно прискорити процес розробки, замінюючи довгі та складні фрагменти коду одним простим і зрозумілим викликом макросу.