**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 MASM»

Варіант 21

Виконав: Перевірив:

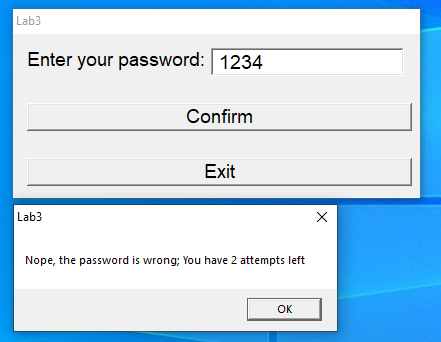
студент групи ІМ-21 доцент Павлов В.Г.

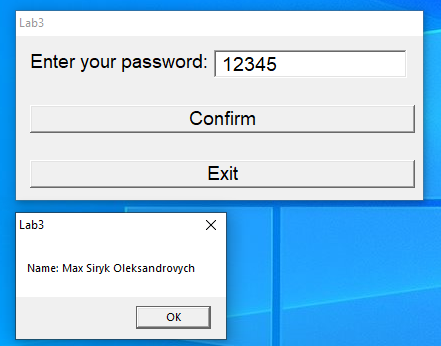
Сірик Максим Олександрович

Номер у списку групи: 21

**Київ 2024**

# Скриншоти виконання програми





# Шифрування паролю

Пароль 12345

Ключ someverystrongkey

1 = 49=> 0011 0001 XOR s = 115 => 0111 0011 = 0110 0010 = B

2 = 50 => 0011 0010 XOR o = 111 => 0110 1111 = 0101 1101 = ]

3 = 51 => 0011 0011 XOR m = 109 => 0110 1101 = 0101 1110 = ^

4 = 52 => 0011 0100 XOR e = 101 => 0110 0101 = 0101 0001 = Q

5 = 53 => 0011 0101 XOR v = 118 => 0111 0110 = 0100 0011 = C

Шифрований пароль B]^QC

# Лістинг

У мене ворд не може витримати такої кількості символів, що у лістингу, також я надішлю Вам файл поштою, тому вставлю лише необхідну частину тексту

### Лістинг 1

00000000 .const

= 000003E9 IDD\_INPUT equ 1001

00000000 .data

00000000 4C 61 62 33 00 MTitle db "Lab3", 0

00000005 4E 61 6D 65 3A my\_name db "Name: Max Siryk Oleksandrovych", 0

20 4D 61 78

20 53 69 72

79 6B 20 4F

6C 65 6B 73

61 6E 64 72

6F 76 79 63

68 00

00000024 42 69 72 74 68 my\_birthday db "Birthday: 17.06.2005", 0

64 61 79 3A

20 31 37 2E

30 36 2E 32

30 30 35 00

00000039 4E 75 6D 62 65 my\_number db "Number: 9056", 0

72 3A 20 39

30 35 36 00

00000046 4E 6F 70 65 2C wrong\_msg\_fmt db "Nope, the password is wrong; You have %d attempts left", 0

20 74 68 65

20 70 61 73

73 77 6F 72

64 20 69 73

20 77 72 6F

6E 67 3B 20

59 6F 75 20

68 61 76 65

20 25 64 20

61 74 74 65

6D 70 74 73

20 6C 65 66

74 00

0000007D 49 6E 70 75 74 prompt\_msg db "Input your password: ", 0

20 79 6F 75

72 20 70 61

73 73 77 6F

72 64 3A 20

00

00000093 42 5D 5E 51 43 pass db "B]^QC", 0

00

00000099 73 6F 6D 65 76 key db "someverystrongkey", 0

65 72 79 73

74 72 6F 6E

67 6B 65 79

00

000000AB 00000003 counter DWORD 3

000000AF 00000100 [ input\_buffer db 256 dup(?)

00

]

000001AF 00000100 [ buff db 256 dup(?)

00

]

my\_show macro info

xor bx, bx ; =============== shows the $info in the messagebox ===============

invoke MessageBox, 0, addr info, addr MTitle, 0 ;; ------------- using default MTitle -------------

endm

my\_cypher macro input, key, output

mov bx, 0

; ------------- making sure bx is 0 -------------

loop\_cypher: ; =============== cyphers input with a key and writes it to the output ===============

.if input\_buffer[bx] == 0

jmp end\_loop\_cypher

.endif

; =============== commenting empty line ===============

mov al, input[bx]

xor al, key[bx]

mov output[bx], al

inc bx

;; ------------- commenting empty line -------------

jmp loop\_cypher

end\_loop\_cypher:

xor bx, bx ; =============== clearing the registers ===============

xor al, al

endm

my\_compare macro first, second

local loop\_label2, right\_label, wrong\_label

mov bx, 0 ; =============== compares the first string with the second ===============

loop\_label:

.if first[bx] == 0 && second[bx] == 0

jmp right\_label

.elseif first[bx] == 0 && second[bx] != 0

jmp wrong\_label

.endif

mov al, first[bx]

mov ah, second[bx]

cmp al, ah

jne wrong\_label

inc bx

jmp loop\_label

right\_label:

call right

jmp exit\_comparison ;; ------------- hidden: making sure to exit -------------

wrong\_label:

call wrong

exit\_comparison:

xor bx, bx

; ------------- clearing the register -------------

endm

00000000 .code

00000000 wrong proc

00000000 FF 0D 000000AB R dec counter

00000006 8B 1D 000000AB R mov ebx, counter

invoke wsprintf, ADDR buff, addr wrong\_msg\_fmt, ebx

my\_show buff

0000001F 66| 33 DB 1 xor bx, bx ; =============== shows the $info in the messagebox ===============

00000035 C3 ret

00000036 wrong endp

00000036 right proc

my\_show my\_name

00000036 66| 33 DB 1 xor bx, bx ; =============== shows the $info in the messagebox ===============

my\_show my\_birthday

0000004C 66| 33 DB 1 xor bx, bx ; =============== shows the $info in the messagebox ===============

my\_show my\_number

00000062 66| 33 DB 1 xor bx, bx ; =============== shows the $info in the messagebox ===============

00000078 C3 ret

00000079 right endp

00000079 DlgProc PROC hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM

.if uMsg == WM\_COMMAND

00000089 8B 45 10 mov eax, wParam

.if ax == IDOK

invoke GetDlgItemText, hWnd, IDD\_INPUT, addr input\_buffer, sizeof input\_buffer

my\_cypher input\_buffer, key, input\_buffer

000000AD 66| BB 0000 1 mov bx, 0

000000B1 1 loop\_cypher: ; =============== cyphers input with a key and writes it to the output ===============

000000BA EB 16 1 jmp end\_loop\_cypher

000000BC 8A 83 000000AF R 1 mov al, input\_buffer[bx]

000000C2 32 83 00000099 R 1 xor al, key[bx]

000000C8 88 83 000000AF R 1 mov input\_buffer[bx], al

000000CE 66| 43 1 inc bx

000000D0 EB DF 1 jmp loop\_cypher

000000D2 1 end\_loop\_cypher:

000000D2 66| 33 DB 1 xor bx, bx ; =============== clearing the registers ===============

000000D5 32 C0 1 xor al, al

my\_compare pass, input\_buffer

000000D7 66| BB 0000 1 mov bx, 0 ; =============== compares the first string with the second ===============

000000DB 1 loop\_label:

000000ED EB 2A 1 jmp ??001A

00000103 EB 1B 1 jmp ??001B

00000105 8A 83 00000093 R 1 mov al, pass[bx]

0000010B 8A A3 000000AF R 1 mov ah, input\_buffer[bx]

00000111 38 E0 1 cmp al, ah

00000113 75 0B 1 jne ??001B

00000115 66| 43 1 inc bx

00000117 EB C2 1 jmp loop\_label

00000119 1 ??001A:

00000119 E8 FFFFFF18 1 call right

0000011E EB 05 1 jmp exit\_comparison

00000120 1 ??001B:

00000120 E8 FFFFFEDB 1 call wrong

00000125 1 exit\_comparison:

00000125 66| 33 DB 1 xor bx, bx

.if counter <= 0

invoke EndDialog, hWnd, IDOK

.endif

.elseif ax == IDCANCEL

invoke EndDialog, hWnd, IDCANCEL

.endif

.elseif uMsg == WM\_CLOSE

invoke EndDialog, hWnd, IDCANCEL

.endif

0000015F 33 C0 xor eax, eax

ret

00000165 DlgProc ENDP

00000165 main:

; dialog macro defined in C:\masm32\include\dialogs

; quoted\_text\_title,quoted\_font,fsize,dstyle,ctlcnt (number of elements),tx,ty,wd,ht,bsize

Dialog "Lab3", "Arial", 14, DS\_CENTER, 4, 0, 0, 150, 70, 1024

# Лістинг 2

include 4-21-IM-21-Siryk-separate-macros.inc

C IFNDEF MY\_MACRO

= 1 C MY\_MACRO equ <1>

C

C my\_show macro info

C xor bx, bx ; =============== shows the $info in the messagebox ===============

C invoke MessageBox, 0, addr info, addr MTitle, 0 ;; ------------- using default MTitle -------------

C endm

C

C

C my\_cypher macro input, key, output

C mov bx, 0

; ------------- making sure bx is 0 -------------

C

C loop\_cypher: ; =============== cyphers input with a key and writes it to the output ===============

C .if input\_buffer[bx] == 0

C jmp end\_loop\_cypher

C .endif

C

C ; =============== commenting empty line ===============

C

C mov al, input[bx]

C xor al, key[bx]

C mov output[bx], al

C inc bx

C ;; ------------- commenting empty line -------------

C jmp loop\_cypher

C

C end\_loop\_cypher:

C xor bx, bx ; =============== clearing the registers ===============

C xor al, al

C endm

C

C

C my\_compare macro first, second

C local loop\_label2, right\_label, wrong\_label

C mov bx, 0 ; =============== compares the first string with the second ===============

C loop\_label:

C

C .if first[bx] == 0 && second[bx] == 0

C jmp right\_label

C .elseif first[bx] == 0 && second[bx] != 0

C jmp wrong\_label

C .endif

C

C mov al, first[bx]

C mov ah, second[bx]

C cmp al, ah

C jne wrong\_label

C

C inc bx

C jmp loop\_label

C

C right\_label:

C call right

C jmp exit\_comparison ;; ------------- hidden: making sure to exit -------------

C

C wrong\_label:

C call wrong

C

C exit\_comparison:

C xor bx, bx

; ------------- clearing the register -------------

C endm

C

C

C ELSE

C ENDIF

C

C

00000000 .const

= 000003E9 IDD\_INPUT equ 1001

00000000 .data

00000000 4C 61 62 33 00 MTitle db "Lab3", 0

00000005 4E 61 6D 65 3A my\_name db "Name: Max Siryk Oleksandrovych", 0

20 4D 61 78

20 53 69 72

79 6B 20 4F

6C 65 6B 73

61 6E 64 72

6F 76 79 63

68 00

00000024 42 69 72 74 68 my\_birthday db "Birthday: 17.06.2005", 0

64 61 79 3A

20 31 37 2E

30 36 2E 32

30 30 35 00

00000039 4E 75 6D 62 65 my\_number db "Number: 9056", 0

72 3A 20 39

30 35 36 00

00000046 4E 6F 70 65 2C wrong\_msg\_fmt db "Nope, the password is wrong; You have %d attempts left", 0

20 74 68 65

20 70 61 73

73 77 6F 72

64 20 69 73

20 77 72 6F

6E 67 3B 20

59 6F 75 20

68 61 76 65

20 25 64 20

61 74 74 65

6D 70 74 73

20 6C 65 66

74 00

0000007D 49 6E 70 75 74 prompt\_msg db "Input your password: ", 0

20 79 6F 75

72 20 70 61

73 73 77 6F

72 64 3A 20

00

00000093 42 5D 5E 51 43 pass db "B]^QC", 0

00

00000099 73 6F 6D 65 76 key db "someverystrongkey", 0

65 72 79 73

74 72 6F 6E

67 6B 65 79

00

000000AB 00000003 counter DWORD 3

000000AF 00000100 [ input\_buffer db 256 dup(?)

00

]

000001AF 00000100 [ buff db 256 dup(?)

00

]

00000000 .code

00000000 wrong proc

00000000 FF 0D 000000AB R dec counter

00000006 8B 1D 000000AB R mov ebx, counter

invoke wsprintf, ADDR buff, addr wrong\_msg\_fmt, ebx

my\_show buff

0000001F 66| 33 DB 1 xor bx, bx ; =============== shows the $info in the messagebox ===============

00000035 C3 ret

00000036 wrong endp

00000036 right proc

my\_show my\_name

00000036 66| 33 DB 1 xor bx, bx ; =============== shows the $info in the messagebox ===============

my\_show my\_birthday

0000004C 66| 33 DB 1 xor bx, bx ; =============== shows the $info in the messagebox ===============

my\_show my\_number

00000062 66| 33 DB 1 xor bx, bx ; =============== shows the $info in the messagebox ===============

00000078 C3 ret

00000079 right endp

00000079 DlgProc PROC hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM

.if uMsg == WM\_COMMAND

00000089 8B 45 10 mov eax, wParam

.if ax == IDOK

invoke GetDlgItemText, hWnd, IDD\_INPUT, addr input\_buffer, sizeof input\_buffer

my\_cypher input\_buffer, key, input\_buffer

000000AD 66| BB 0000 1 mov bx, 0

000000B1 1 loop\_cypher: ; =============== cyphers input with a key and writes it to the output ===============

000000BA EB 16 1 jmp end\_loop\_cypher

000000BC 8A 83 000000AF R 1 mov al, input\_buffer[bx]

000000C2 32 83 00000099 R 1 xor al, key[bx]

000000C8 88 83 000000AF R 1 mov input\_buffer[bx], al

000000CE 66| 43 1 inc bx

000000D0 EB DF 1 jmp loop\_cypher

000000D2 1 end\_loop\_cypher:

000000D2 66| 33 DB 1 xor bx, bx ; =============== clearing the registers ===============

000000D5 32 C0 1 xor al, al

my\_compare pass, input\_buffer

000000D7 66| BB 0000 1 mov bx, 0 ; =============== compares the first string with the second ===============

000000DB 1 loop\_label:

000000ED EB 2A 1 jmp ??001A

00000103 EB 1B 1 jmp ??001B

00000105 8A 83 00000093 R 1 mov al, pass[bx]

0000010B 8A A3 000000AF R 1 mov ah, input\_buffer[bx]

00000111 38 E0 1 cmp al, ah

00000113 75 0B 1 jne ??001B

00000115 66| 43 1 inc bx

00000117 EB C2 1 jmp loop\_label

00000119 1 ??001A:

00000119 E8 FFFFFF18 1 call right

0000011E EB 05 1 jmp exit\_comparison

00000120 1 ??001B:

00000120 E8 FFFFFEDB 1 call wrong

00000125 1 exit\_comparison:

00000125 66| 33 DB 1 xor bx, bx

.if counter <= 0

invoke EndDialog, hWnd, IDOK

.endif

.elseif ax == IDCANCEL

invoke EndDialog, hWnd, IDCANCEL

.endif

.elseif uMsg == WM\_CLOSE

invoke EndDialog, hWnd, IDCANCEL

.endif

0000015F 33 C0 xor eax, eax

ret

00000165 DlgProc ENDP

00000165 main:

; dialog macro defined in C:\masm32\include\dialogs

; quoted\_text\_title,quoted\_font,fsize,dstyle,ctlcnt (number of elements),tx,ty,wd,ht,bsize

Dialog "Lab3", "Arial", 14, DS\_CENTER, 4, 0, 0, 150, 70, 1024

# Дослідження

Отже, при визначенні самих макросів, ми можемо поачити, що їхній зміст повністю зберігається.

При їхній трансляції зникають коментарі, що були визначені як “;;”, що відповідає теорії “;; - символ, що пригнічує виділення памяті для коментарю в макровизначенні”, - . Рисований О.М. Системне програмування. – Харків, : НТУ «Харківський політехнічний інститут», 2012, 912 с. ст. 74. Варто зауважити, що коментарі, які не стосуються лексикології компілятора зникають, не важливо, чи вони приховані, чи ні (commenting empty line зникло)

Також транслюються invoke (MessageBox), та інші макроси, такі як .IF. На їх місцізявляється “66| 33 DB”, (??001A) можна припустити, що це вказівники на память.

Якщо порівнювати файл із підключенням макросів із самаго файлу із файлом підключеними макросами із зовнішнього файлу, то можна помітити відсутність відмінності між транслюваним кодом, однак все ж таки є можливість визначити, що ми підключали макроси із зовнішнього файлу, адже файл із зовнішнім підклученням макросів має “C” перед початком рядкув, а також йдє після директиви include, що власне і містить інформацію звідки нам підключати файл, навідмінну від першого випадку, де макроси були визначені де ми їх і прописали.

У ході лаб. Роботи, я також визначав мітки (labels). Звичайні мітки залишаються без змін (тобто мітка loop\_cypher, залишиться такою ж), однак локальні мітки, як і очікувалися, матимуть унікальні імена (??0019), можна припустити, що це адреса.

# Висновок

Отже, в ході лабораторної роботи № 4 я ознайомився з макросами. Дослідивши зсередини як працюють макроси, тобто дослідивши розширений лістинг, я дізнався, що при визначенні макросів вони зберагаються тік як є, а вже при трансляції зникають певні елементи, що наведені у секції вище. Також якщо ми імпортуємо макроси вони у лістингу йдуть після директиви include. Також проаназіувавши лістинг можна дійти до висновку, що я використовував вже вбудовані макроси також. Локальні мітки, як і наведено в секції вище, мають унікальні імена. Щодо “прихованих кментарів”, то на мою думку, це невдала назва, адже це символ що пригнічює виділення памяті для коментарю, і вже як наслідок текст зникає, хоча частково сенс зрозумілий. Як показали експерименти, символ “;;” працює безвідмовно, та текст зникає