**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  
«Технології розроблення системних програм»

на тему  
«Макровизначення і макроси в MASM32.»

Виконав: Перевірив:

студент групи ІП-84 доц. кафедри ОТ  
Голубов Іван Олегович Павлов В. Г.  
номер залікової книжки: 8404

Київ 2020

**Мета роботи**

Вивчити технології створення і вживання макросів. Дослідження результатів роботи макрогенератору MASM32.

**Порядок виконання роботи**

1. Вивчити правила використання макросів в програмах [1].

2. Переробити програму, підготовлену в лабораторній роботі 6 так, щоб виведення кожного виду інформації (дата народження, номер залікової книжки і т. п.) виконувалося у вигляді багатократного звернення до макросу (макрос 1), в якому у віконному інтерфейсі здійснювалося б виведення тексту, який передається в макрос як параметр.

3. Оформити також у вигляді окремих макросів фрагменти програми, де здійснюється шифрування введеного рядка символів (макрос 2), а також порівняння її з хеш-кодом оригіналу пароля, що зберігається у програмі (макрос 3). Макровизначення для всіх макросів розмістити в тому ж файлі, що і програма. У всіх макросах обов'язково використовувати звичайні і приховані коментарі, а в останньому макросі також використовувати механізм оголошення локальних міток.

4. Виконати компіляцію і компоновку файлу програми.

5. Перевірити роботу програми шляхом введення як правильного, так і невірного паролів.

6. Отримати розширений лістинг програми за допомогою опції /Fl компілятора ML.

7. Провести дослідження отриманого лістингу: визначити, яким чином компілятор виконав трансляцію кожній з команд макросів і помітити їх в тексті розширеного лістингу кольоровими олівцями або кольоровими фломастерами.

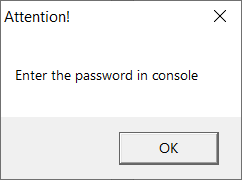
8. Повторити дослідження, які виконані в п. 4 – 7 для другої версії програми, де всі перераховані в п. 2 і 3 макровизначення для макросів повинні розташовуватися в окремому файлі.

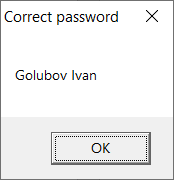
9. Порівняти тексти розширених лістингів для обох варіантів розміщення макровизначень, знайти схожість і відмінності. Відобразити їх в звіті по лабораторній роботі.

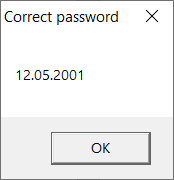
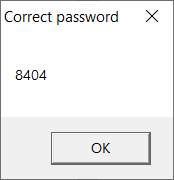
10. Зробити висновки по лабораторній роботі.

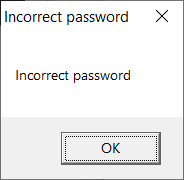
**Скріншоти**

**Програма в 1 файлі**

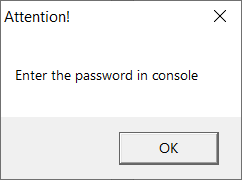


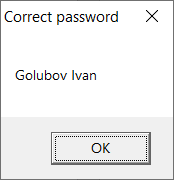
 

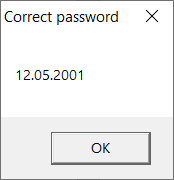
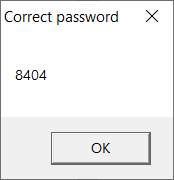
 

**Програма (в окремих файлах)**





**Вихідний код**

**Програма з макросами в 1 файлі**

.686

include c:\masm32\include\masm32rt.inc

includelib \masm32\lib\user32.lib

.data

;///////////////////////////////////////////////////////////////////////////

; Data /

;///////////////////////////////////////////////////////////////////////////

myName db "Golubov Ivan", 0

myDate db "12.05.2001", 0

myNumb db "8404", 0

inputb db 128 dup (0), '$'

hConsoleInput dd 0

ConsoleLen db 0

message0 db "Enter the password in console",0

message1 db "Correct password",10, 13, 0

message2 db "Incorrect password", 0

message3 db "Attention!",0

caption db "Input", 0

password db 3Ah, 36h, 32h, 36h

PassLen=$-password

key equ 02h

MOutput macro info, msg ;; macros for output

invoke MessageBox, 0, offset info, offset msg ,MB\_OK

endm

MDecrypt macro key, len ; macros for encryption

xor esi, esi

hash:

xor password[esi], key

inc esi

cmp esi, len ; if len!=4 => hash

jne hash

endm

MCompare macro input, pass, len ;; macros for comparing password

local righ

lea ebx, input

call stringlength

mov edi, esi

mov edi, len

sub edi, esi

cmp edi, 0

jne wrng

righ:

cld

mov ecx, len

lea esi, input

lea edi, pass

repe cmpsb

jne wrng

endm

.code

start:

MDecrypt key, PassLen

MOutput message0, message3

call AllocConsole

invoke SetConsoleTitle, offset caption

invoke GetStdHandle, STD\_INPUT\_HANDLE

mov hConsoleInput, eax

invoke ReadConsole, hConsoleInput,offset inputb, 128, offset ConsoleLen, NULL

MCompare inputb, password, PassLen

MOutput myName, message1

MOutput myDate, message1

MOutput myNumb, message1

invoke ExitProcess, 0

wrng proc

MOutput message2, message2

invoke ExitProcess, 0

wrng endp

;///////////////////////////////////////////////////////////////////////////

stringlength proc

mov esi, 0

mov edx, 0

iter:

mov eax, [ebx+esi]

inc esi

cmp eax, edx

jne iter

sub esi, 3

ret

stringlength endp

;///////////////////////////////////////////////////////////////////////////

end start

**Програма (в окремих файлах)**

.686

include c:\masm32\include\masm32rt.inc

includelib \masm32\lib\user32.lib

include macros.inc

.data

myName db "Golubov Ivan", 0

myDate db "12.05.2001", 0

myNumb db "8404", 0

inputb db 128 dup (0), '$'

hConsoleInput dd 0

ConsoleLen db 0

message0 db "Enter the password in console",0

message1 db "Correct password",10, 13, 0

message2 db "Incorrect password", 0

message3 db "Attention!",0

caption db "Input", 0

password db 3Ah, 36h, 32h, 36h

PassLen=$-password

key equ 02h

.code

start:

MDecrypt key, PassLen

MOutput message0, message3

call AllocConsole

invoke SetConsoleTitle, offset caption

invoke GetStdHandle, STD\_INPUT\_HANDLE

mov hConsoleInput, eax

invoke ReadConsole, hConsoleInput,offset inputb, 128, offset ConsoleLen, NULL

MCompare inputb, password, PassLen

MOutput myName, message1

MOutput myDate, message1

MOutput myNumb, message1

invoke ExitProcess, 0

wrng proc

MOutput message2, message2

invoke ExitProcess, 0

wrng endp

stringlength proc

mov esi, 0

mov edx, 0

iter:

mov eax, [ebx+esi]

inc esi

cmp eax, edx

jne iter

sub esi, 3

ret

stringlength endp

end start

**Макроси**

MOutput macro info, msg ;; macros for output

invoke MessageBox, 0, offset info, offset msg ,MB\_OK

endm

MDecrypt macro key, len ; macros for encryption

xor esi, esi

hash:

xor password[esi], key

inc esi

cmp esi, len ; if len!=4 => hash

jne hash

endm

MCompare macro input, pass, len ;; macros for comparing password

local righ

lea ebx, input

call stringlength

mov edi, esi

mov edi, len

sub edi, esi

cmp edi, 0

jne wrng

righ:

cld

mov ecx, len

lea esi, input

lea edi, pass

repe cmpsb

jne wrng

endm

**Розширений лістинг програми в 1 файлі**

includelib \masm32\lib\user32.lib

00000000 .data

;///////////////////////////////////////////////////////////////////////////

; Data /

;///////////////////////////////////////////////////////////////////////////

00000000 47 6F 6C 75 62 myName db "Golubov Ivan", 0

6F 76 20 49

76 61 6E 00

0000000D 31 32 2E 30 35 myDate db "12.05.2001", 0

2E 32 30 30

31 00

00000018 38 34 30 34 00 myNumb db "8404", 0

0000001D 24 inputb db 128 dup (0), '$'

0000009E 00000000 hConsoleInput dd 0

000000A2 00 ConsoleLen db 0

000000A3 45 6E 74 65 72 message0 db "Enter the password in console",0

20 74 68 65

20 70 61 73

73 77 6F 72

64 20 69 6E

20 63 6F 6E

73 6F 6C 65

00

000000C1 43 6F 72 72 65 message1 db "Correct password",10, 13, 0

63 74 20 70

61 73 73 77

6F 72 64 0A

0D 00

000000D4 49 6E 63 6F 72 message2 db "Incorrect password", 0

72 65 63 74

20 70 61 73

73 77 6F 72

64 00

000000E7 41 74 74 65 6E message3 db "Attention!",0

74 69 6F 6E

21 00

000000F2 49 6E 70 75 74 caption db "Input", 0

00

000000F8 3A 36 32 36 password db 3Ah, 36h, 32h, 36h

000000FC = 00000004 PassLen=$-password

= 00000002 key equ 02h

MOutput macro info, msg ;; macros for output

invoke MessageBox, 0, offset info, offset msg ,MB\_OK

endm

MDecrypt macro key, len ; macros for encryption

xor esi, esi

hash:

xor password[esi], key

inc esi

cmp esi, len ; if len!=4 => hash

jne hash

endm

MCompare macro input, pass, len ;; macros for comparing password

local righ

lea ebx, input

call stringlength

mov edi, esi

mov edi, len

sub edi, esi

cmp edi, 0

jne wrng

righ:

cld

mov ecx, len

lea esi, input

lea edi, pass

repe cmpsb

jne wrng

endm

00000000 .code

00000000 start:

MDecrypt key, PassLen

00000000 33 F6 1 xor esi, esi

00000002 1 hash:

00000002 80 B6 000000F8 R 1 xor password[esi], key

02

00000009 46 1 inc esi

0000000A 81 FE 00000004 1 cmp esi, PassLen ; if len!=4 => hash

00000010 75 F0 1 jne hash

MOutput message0, message3

00000025 E8 00000000 E call AllocConsole

invoke SetConsoleTitle, offset caption

invoke GetStdHandle, STD\_INPUT\_HANDLE

0000003B A3 0000009E R mov hConsoleInput, eax

invoke ReadConsole, hConsoleInput,offset inputb, 128, offset ConsoleLen, NULL

MCompare inputb, password, PassLen

0000005C 8D 1D 0000001D R 1 lea ebx, inputb

00000062 E8 0000007E 1 call stringlength

00000067 8B FE 1 mov edi, esi

00000069 BF 00000004 1 mov edi, PassLen

0000006E 2B FE 1 sub edi, esi

00000070 83 FF 00 1 cmp edi, 0

00000073 75 56 1 jne wrng

00000075 1 ??0019:

00000075 FC 1 cld

00000076 B9 00000004 1 mov ecx, PassLen

0000007B 8D 35 0000001D R 1 lea esi, inputb

00000081 8D 3D 000000F8 R 1 lea edi, password

00000087 F3/ A6 1 repe cmpsb

00000089 75 40 1 jne wrng

MOutput myName, message1

MOutput myDate, message1

MOutput myNumb, message1

invoke ExitProcess, 0

000000CB wrng proc

MOutput message2, message2

invoke ExitProcess, 0

000000E5 wrng endp

;///////////////////////////////////////////////////////////////////////////

000000E5 stringlength proc

000000E5 BE 00000000 mov esi, 0

000000EA BA 00000000 mov edx, 0

000000EF iter:

000000EF 8B 04 1E mov eax, [ebx+esi]

000000F2 46 inc esi

000000F3 3B C2 cmp eax, edx

000000F5 75 F8 jne iter

000000F7 83 EE 03 sub esi, 3

000000FA C3 ret

000000FB stringlength endp

;///////////////////////////////////////////////////////////////////////////

end start

**Опис розміщення макровизначень між кодом і роширеним лістингом**

Відміність полягає у тому, що включені в макрос данні дублюються при подальшому виклику макросу.

**Розширений лістинг програми з макросами в окремих файлах**

**Програма**

includelib \masm32\lib\user32.lib

include macros.inc

C MOutput macro info, msg ;; macros for output

C invoke MessageBox, 0, offset info, offset msg ,MB\_OK

C endm

C

C MDecrypt macro key, len ; macros for encryption

C xor esi, esi

C hash:

C xor password[esi], key

C inc esi

C cmp esi, len ; if len!=4 => hash

C jne hash

C endm

C

C MCompare macro input, pass, len ;; macros for comparing password

C local righ

C lea ebx, input

C call stringlength

C

C mov edi, esi

C

C mov edi, len

C sub edi, esi

C cmp edi, 0

C jne wrng

C

C righ:

C cld

C mov ecx, len

C lea esi, input

C lea edi, pass

C repe cmpsb

C jne wrng

C endm

C

00000000 .data

00000000 47 6F 6C 75 62 myName db "Golubov Ivan", 0

6F 76 20 49

76 61 6E 00

0000000D 31 32 2E 30 35 myDate db "12.05.2001", 0

2E 32 30 30

31 00

00000018 38 34 30 34 00 myNumb db "8404", 0

0000001D 24 inputb db 128 dup (0), '$'

0000009E 00000000 hConsoleInput dd 0

000000A2 00 ConsoleLen db 0

000000A3 45 6E 74 65 72 message0 db "Enter the password in console",0

20 74 68 65

20 70 61 73

73 77 6F 72

64 20 69 6E

20 63 6F 6E

73 6F 6C 65

00

000000C1 43 6F 72 72 65 message1 db "Correct password",10, 13, 0

63 74 20 70

61 73 73 77

6F 72 64 0A

0D 00

000000D4 49 6E 63 6F 72 message2 db "Incorrect password", 0

72 65 63 74

20 70 61 73

73 77 6F 72

64 00

000000E7 41 74 74 65 6E message3 db "Attention!",0

74 69 6F 6E

21 00

000000F2 49 6E 70 75 74 caption db "Input", 0

00

000000F8 3A 36 32 36 password db 3Ah, 36h, 32h, 36h

000000FC = 00000004 PassLen=$-password

= 00000002 key equ 02h

00000000 .code

00000000 start:

MDecrypt key, PassLen

00000000 33 F6 1 xor esi, esi

00000002 1 hash:

00000002 80 B6 000000F8 R 1 xor password[esi], key

02

00000009 46 1 inc esi

0000000A 81 FE 00000004 1 cmp esi, PassLen ; if len!=4 => hash

00000010 75 F0 1 jne hash

MOutput message0, message3

00000025 E8 00000000 E call AllocConsole

invoke SetConsoleTitle, offset caption

invoke GetStdHandle, STD\_INPUT\_HANDLE

0000003B A3 0000009E R mov hConsoleInput, eax

invoke ReadConsole, hConsoleInput,offset inputb, 128, offset ConsoleLen, NULL

MCompare inputb, password, PassLen

0000005C 8D 1D 0000001D R 1 lea ebx, inputb

00000062 E8 0000007E 1 call stringlength

00000067 8B FE 1 mov edi, esi

00000069 BF 00000004 1 mov edi, PassLen

0000006E 2B FE 1 sub edi, esi

00000070 83 FF 00 1 cmp edi, 0

00000073 75 56 1 jne wrng

00000075 1 ??0019:

00000075 FC 1 cld

00000076 B9 00000004 1 mov ecx, PassLen

0000007B 8D 35 0000001D R 1 lea esi, inputb

00000081 8D 3D 000000F8 R 1 lea edi, password

00000087 F3/ A6 1 repe cmpsb

00000089 75 40 1 jne wrng

MOutput myName, message1

MOutput myDate, message1

MOutput myNumb, message1

invoke ExitProcess, 0

000000CB wrng proc

MOutput message2, message2

invoke ExitProcess, 0

000000E5 wrng endp

000000E5 stringlength proc

000000E5 BE 00000000 mov esi, 0

000000EA BA 00000000 mov edx, 0

000000EF iter:

000000EF 8B 04 1E mov eax, [ebx+esi]

000000F2 46 inc esi

000000F3 3B C2 cmp eax, edx

000000F5 75 F8 jne iter

000000F7 83 EE 03 sub esi, 3

000000FA C3 ret

000000FB stringlength endp

end start

**Опис розміщення макровизначень між 2 лістингами**

Відмінне в розміщені макросів цих 2 програм це те, що оголошення макросу для розділеного файлу відбувається після підключення бібліотек на відміну з макросами в 1 файлі, які оголошенні в тому місці де їх розмістили. Інших відмінностей не знайдено.

**Висновки**

У ході виконання лабораторної роботи я вивчив технології створення і вживання макросів й дослідив результатів роботи макрогенератору MASM32.

Я створив програму на мові асемблер, в якій за допомогою макросів шифрую пароль, порівнюю введені дані з паролем та виводжу певну інформацію на екран. Дослідив виконання програми при введенні правильного та неправильного паролю. Потім я виніс створені макроси в окремий файл inc, й включив його до основного файлу за допомогою include. Порівняв розширений лістинг цих двох варіантів програм, і визначив що вони відрізняються лише місцем розташування макросів.

Зрозумів, що включені в макрос данні дублюються при подальшому виклику макросу, а також, що оголошення макросу в програмі з файлом inc відбувається на початку програми, на відміну від з’єднаної програми.