Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Обчислювальної техніки

Лабораторна робота № 4

з дисципліни «Системне програмування»

на тему

«Макровизначення і макроси в MASM32»

Виконала:

студентка гр. ІП-93

Узунлу Гаизенур

Викладач:

доц. Павлов В. Г.

Київ – 2021

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

**Програма 1. Код програми з макросами, визначеними всередині**

.386

.model flat, stdcall

option casemap :none

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Imports \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

include c:/masm32/include/windows.inc

include c:/masm32/include/user32.inc

include c:/masm32/include/kernel32.inc

includelib c:/masm32/lib/user32.lib

includelib c:/masm32/lib/kernel32.lib

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Macroses \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printLabel macro printedText, xPos:=<LabelXPos>, yPos:=<LabelYPos>

; Create label using passed parameters

;; This comment won't be included to listing file

COMMENT \*

The comment directive.

Ends on the symbol, that goes after word COMMENT.

\*

invoke CreateWindowEx, NULL,

addr LabelField, addr printedText,

WS\_VISIBLE or WS\_CHILD,

xPos, yPos, LabelWidth, LabelHeight,

hWnd, LabelID, hInstance, NULL

endm

cipherText macro ciphered

;; Declare local labels

local cycle

local endPoint

; Clear di register (it can be full with a value)

mov di, 0

cycle:

cmp di, len

je endPoint

mov dh, key

xor ciphered[di], dh

inc di

jne cycle

endPoint:

endm

compareText macro ciphered, passwordText

local cycle

local endPoint

mov di, 0

cycle:

cmp di, len

je endPoint

; If we didn't reach the end, check

mov dh, ciphered[di]

cmp dh, passwordText[di]

; If the letters aren't equal, end of comparing

jne endPoint

inc di

jmp cycle

;; Comment test

endPoint:

endm

;\*\*\*\*\*\*\*\*\*\*\*\*\*\* Procedure prototypes \*\*\*\*\*\*\*\*\*\*\*\*\*\*

WinMain proto :dword, :dword, :dword, :dword

WndProc proto :dword, :dword, :dword, :dword

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.data

; Window properties

WndTitle DB "Log In Window", 0

WndXPos equ 20

WndYPos equ 20

WndHeight equ 240

WndWidth equ 300

; Label properties

LabelField db "static", 0

LabelText db "Enter the password", 0

LabelID equ 2000

LabelXPos equ 20

LabelYPos equ 20

LabelHeight equ 20

LabelWidth equ 160

; Editing field properties

EditField db "edit", 0

EditID equ 2001

EditXPos equ 20

EditYPos equ 50

EditHeight equ 20

EditWidth equ 160

; Button properties

Button db "button", 0

ButtonText db "Try", 0

ButtonID equ 2002

ButtonXPos equ 200

ButtonYPos equ 20

ButtonHeight equ 20

ButtonWidth equ 70

; Exit button

ExitButtonText db "Exit", 0

ExitButtonID equ 2003

ExitButtonXPos equ 200

ExitButtonYPos equ 50

ExitButtonHeight equ 20

ExitButtonWidth equ 70

; Specific parameters

KeyboardHookWParam equ 40

MsgBoxTitle db "Message", 0

WndClassName DB "BasicWindow", 0

; Error message related data

ErrorMessageOne db "The input is wrong.", 0

ErrorMessageTwo db "Try again.", 0

ErrorMessageThree db "I believe in you)", 0

; Student related data

nameText db "Uzunlu Gamzenur", 0

nameTextXPos equ 40

nameTextYPos equ 90

numberText db "9882", 0

numberTextXPos equ 40

numberTextYPos equ 120

dateText db "11.10.1999", 0

dateTextXPos equ 40

dateTextYPos equ 150

; The password is "abcdefgh"

password db "dgfa`cbm"

key db 00000101b

len dw 8

.data?

hInstance dd ?

lpCmdLine dd ?

hEditText HWND ?

input db 64 DUP (?)

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.code

start:

; Invoke GUI hooks

invoke GetModuleHandle, NULL

mov hInstance, eax

invoke GetCommandLine

mov lpCmdLine, eax

; Launch main window

invoke WinMain, hInstance, NULL, lpCmdLine, SW\_SHOWDEFAULT

; After it finishes, exit

invoke ExitProcess, eax

WinMain proc hInst :dword, hPrevInst :dword, CmdLine :dword, nShowCmd :dword

; WC - window context

local wc :WNDCLASSEX

; Msg - message

local msg :MSG

local hWnd :HWND

; Apply styles to the window

mov wc.cbSize, sizeof WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW or CS\_BYTEALIGNWINDOW

mov wc.lpfnWndProc, WndProc

mov wc.cbClsExtra, NULL

mov wc.cbWndExtra, NULL

; Apply instance

push hInst

pop wc.hInstance

; Apply more window properties

mov wc.hbrBackground, COLOR\_BTNFACE + 1

mov wc.lpszMenuName, NULL

mov wc.lpszClassName, offset WndClassName

; Initialize app icon

invoke LoadIcon, hInst, IDI\_APPLICATION

mov wc.hIcon, eax

mov wc.hIconSm, eax

; Initialize cursor hooks

invoke LoadCursor, hInst, IDC\_ARROW

mov wc.hCursor, eax

; Register window context as a new window

invoke RegisterClassEx, addr wc

; Create GUI window representation

invoke CreateWindowEx, WS\_EX\_APPWINDOW, addr WndClassName, addr WndTitle,

WS\_OVERLAPPEDWINDOW,

WndXPos, WndYPos, WndWidth, WndHeight,

NULL, NULL, hInst, NULL

; Register creation into eax register

mov hWnd, eax

; Show window and update every time it changes

invoke ShowWindow, hWnd, nShowCmd

invoke UpdateWindow, hWnd

; Event emitting

EventEmitter:

invoke GetMessage, addr msg, NULL, 0, 0

cmp eax, 0

je EndOfEvents

invoke TranslateMessage, addr msg

invoke DispatchMessage, addr msg

jmp EventEmitter

EndOfEvents:

mov eax, msg.wParam

ret

WinMain endp

; Window events processor

WndProc proc hWnd :dword, uMsg :dword, wParam :dword, lParam :dword

; If the current event is creating the window

.if uMsg==WM\_CREATE

; Create edit field

invoke CreateWindowEx, NULL,

addr EditField, NULL,

WS\_VISIBLE or WS\_CHILD or ES\_LEFT or ES\_AUTOHSCROLL or ES\_AUTOVSCROLL or WS\_BORDER,

EditXPos, EditYPos, EditWidth, EditHeight,

hWnd, EditID, hInstance, NULL

mov hEditText, eax

; Create label field

printLabel LabelText

; Create "Try" button

invoke CreateWindowEx, NULL,

addr Button, addr ButtonText,

WS\_VISIBLE or WS\_CHILD,

ButtonXPos, ButtonYPos, ButtonWidth, ButtonHeight,

hWnd, ButtonID, hInstance, NULL

; Create "Exit" button

invoke CreateWindowEx, NULL,

addr Button, addr ExitButtonText,

WS\_VISIBLE or WS\_CHILD,

ExitButtonXPos, ExitButtonYPos, ExitButtonWidth, ExitButtonHeight,

hWnd, ExitButtonID, hInstance, NULL

; If the event is the window's command

; (button click, whatever)

.elseif uMsg==WM\_COMMAND

; If clicked "Exit", exit

cmp wParam, ExitButtonID

je stopProgram

; If clicked "Try", start checking the password

cmp wParam, ButtonID

jne stop

; Take out the string inside edit field

invoke SendMessage, hEditText, WM\_GETTEXT,

KeyboardHookWParam, addr input

; If the length of input and password aren't equal,

; there is an error

cmp ax, len

jne inputError

; Otherwise, compare input with password

cipherText input

compareText input, password

; If everything is correct -- zero (Z) flag is set to 0

; Otherwise (are not equal) -- Z flag is is set to 1

je inputSuccess

inputError:

printLabel ErrorMessageOne, nameTextXPos, nameTextYPos

printLabel ErrorMessageTwo, numberTextXPos, numberTextYPos

printLabel ErrorMessageThree, dateTextXPos, dateTextYPos

jmp stop

; Success dialog

inputSuccess:

printLabel nameText, nameTextXPos, nameTextYPos

printLabel numberText, numberTextXPos, numberTextYPos

printLabel dateText, dateTextXPos, dateTextYPos

.elseif uMsg == WM\_DESTROY

stopProgram:

invoke PostQuitMessage, 0

jmp stop

.else

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.endif

; Leave the procedure

stop:

xor eax,eax

ret

WndProc endp

end start

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Компіляція програми відбувається за допомогою скрипту:

if not exist %1.asm (

echo "No such file."

exit /b 1

)

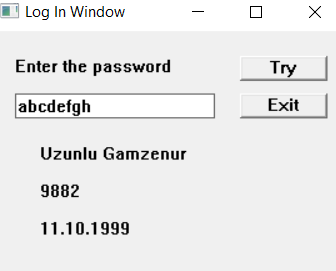
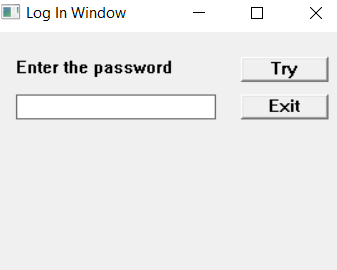
echo "Compiling..."

C:\masm32\bin\ml.exe -c -Zi -Fl -nologo %1.asm

echo "Linking..."

C:\masm32\bin\link.exe /MAP /SUBSYSTEM:CONSOLE %1.obj

Перевіримо на коректність нашу програму:



Таким чином, все працює правильно.

Дослідимо лістинг програми (а саме частину з макросами):

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Macroses \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

printLabel macro printedText, xPos:=<LabelXPos>, yPos:=<LabelYPos>

; Create label using passed parameters

; This comment won't be included to listing file

COMMENT \*

The comment directive.

Ends on the symbol, that goes after word COMMENT.

\*

invoke CreateWindowEx, NULL,

addr LabelField, addr printedText,

WS\_VISIBLE or WS\_CHILD,

xPos, yPos, LabelWidth, LabelHeight,

hWnd, LabelID, hInstance, NULL

endm

cipherText macro ciphered

; Declare local labels

local cycle

local endPoint

; Clear di register (it can be full with a value)

mov di, 0

cycle:

cmp di, len

je endPoint

mov dh, key

xor ciphered[di], dh

inc di

jne cycle

endPoint:

endm

compareText macro ciphered, passwordText

local cycle

local endPoint

mov di, 0

cycle:

cmp di, len

je endPoint

; If we didn't reach the end, check

mov dh, ciphered[di]

cmp dh, passwordText[di]

; If the letters aren't equal, end of comparing

jne endPoint

inc di

jmp cycle

; Comment test

endPoint:

endm

Жовтим позначено коментарі, які б, здавалось би, не мали б з’являтись у файлі лістингу. Причому ці коментарі починаються з одного знаку “крапка з комою”, але у сирцевому коді починається з двох таких знаків.

Загляньмо на самий початок лістингу:

Microsoft (R) Macro Assembler Version 12.00.21005.1

Радше за все, поведінка компілятора з часом змінилась, і тепер такі коментарі включаються у файл лістингу. Згадаймо тих сміливців, які писали після ;; лайливі слова.

Тепер відокремимо код та макроси.

**Програма 2. Код програми з макросами, визначеними в окремому файлі**

.386

.model flat, stdcall

option casemap :none

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Imports \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

include c:/masm32/include/windows.inc

include c:/masm32/include/user32.inc

include c:/masm32/include/kernel32.inc

includelib c:/masm32/lib/user32.lib

includelib c:/masm32/lib/kernel32.lib

include 4-27-IP-93-Uzunlu.inc

;\*\*\*\*\*\*\*\*\*\*\*\*\*\* Procedure prototypes \*\*\*\*\*\*\*\*\*\*\*\*\*\*

WinMain proto :dword, :dword, :dword, :dword

WndProc proto :dword, :dword, :dword, :dword

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.data

; Window properties

WndTitle DB "Log In Window", 0

WndXPos equ 20

WndYPos equ 20

WndHeight equ 240

WndWidth equ 300

; Label properties

LabelField db "static", 0

LabelText db "Enter the password", 0

LabelID equ 2000

LabelXPos equ 20

LabelYPos equ 20

LabelHeight equ 20

LabelWidth equ 160

; Editing field properties

EditField db "edit", 0

EditID equ 2001

EditXPos equ 20

EditYPos equ 50

EditHeight equ 20

EditWidth equ 160

; Button properties

Button db "button", 0

ButtonText db "Try", 0

ButtonID equ 2002

ButtonXPos equ 200

ButtonYPos equ 20

ButtonHeight equ 20

ButtonWidth equ 70

; Exit button

ExitButtonText db "Exit", 0

ExitButtonID equ 2003

ExitButtonXPos equ 200

ExitButtonYPos equ 50

ExitButtonHeight equ 20

ExitButtonWidth equ 70

; Specific parameters

KeyboardHookWParam equ 40

MsgBoxTitle db "Message", 0

WndClassName DB "BasicWindow", 0

; Error message related data

ErrorMessageOne db "The input is wrong.", 0

ErrorMessageTwo db "Try again.", 0

ErrorMessageThree db "I believe in you)", 0

; Student related data

nameText db "Uzunlu Gamzenur", 0

nameTextXPos equ 40

nameTextYPos equ 90

numberText db "9882", 0

numberTextXPos equ 40

numberTextYPos equ 120

dateText db "11.10.1999", 0

dateTextXPos equ 40

dateTextYPos equ 150

; The password is "abcdefgh"

password db "dgfa`cbm"

key db 00000101b

len dw 8

.data?

hInstance dd ?

lpCmdLine dd ?

hEditText HWND ?

input db 64 DUP (?)

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.code

start:

; Invoke GUI hooks

invoke GetModuleHandle, NULL

mov hInstance, eax

invoke GetCommandLine

mov lpCmdLine, eax

; Launch main window

invoke WinMain, hInstance, NULL, lpCmdLine, SW\_SHOWDEFAULT

; After it finishes, exit

invoke ExitProcess, eax

WinMain proc hInst :dword, hPrevInst :dword, CmdLine :dword, nShowCmd :dword

; WC - window context

local wc :WNDCLASSEX

; Msg - message

local msg :MSG

local hWnd :HWND

; Apply styles to the window

mov wc.cbSize, sizeof WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW or CS\_BYTEALIGNWINDOW

mov wc.lpfnWndProc, WndProc

mov wc.cbClsExtra, NULL

mov wc.cbWndExtra, NULL

; Apply instance

push hInst

pop wc.hInstance

; Apply more window properties

mov wc.hbrBackground, COLOR\_BTNFACE + 1

mov wc.lpszMenuName, NULL

mov wc.lpszClassName, offset WndClassName

; Initialize app icon

invoke LoadIcon, hInst, IDI\_APPLICATION

mov wc.hIcon, eax

mov wc.hIconSm, eax

; Initialize cursor hooks

invoke LoadCursor, hInst, IDC\_ARROW

mov wc.hCursor, eax

; Register window context as a new window

invoke RegisterClassEx, addr wc

; Create GUI window representation

invoke CreateWindowEx, WS\_EX\_APPWINDOW, addr WndClassName, addr WndTitle,

WS\_OVERLAPPEDWINDOW,

WndXPos, WndYPos, WndWidth, WndHeight,

NULL, NULL, hInst, NULL

; Register creation into eax register

mov hWnd, eax

; Show window and update every time it changes

invoke ShowWindow, hWnd, nShowCmd

invoke UpdateWindow, hWnd

; Event emitting

EventEmitter:

invoke GetMessage, addr msg, NULL, 0, 0

cmp eax, 0

je EndOfEvents

invoke TranslateMessage, addr msg

invoke DispatchMessage, addr msg

jmp EventEmitter

EndOfEvents:

mov eax, msg.wParam

ret

WinMain endp

; Window events processor

WndProc proc hWnd :dword, uMsg :dword, wParam :dword, lParam :dword

; If the current event is creating the window

.if uMsg==WM\_CREATE

; Create edit field

invoke CreateWindowEx, NULL,

addr EditField, NULL,

WS\_VISIBLE or WS\_CHILD or ES\_LEFT or ES\_AUTOHSCROLL or ES\_AUTOVSCROLL or WS\_BORDER,

EditXPos, EditYPos, EditWidth, EditHeight,

hWnd, EditID, hInstance, NULL

mov hEditText, eax

; Create label field

printLabel LabelText

; Create "Try" button

invoke CreateWindowEx, NULL,

addr Button, addr ButtonText,

WS\_VISIBLE or WS\_CHILD,

ButtonXPos, ButtonYPos, ButtonWidth, ButtonHeight,

hWnd, ButtonID, hInstance, NULL

; Create "Exit" button

invoke CreateWindowEx, NULL,

addr Button, addr ExitButtonText,

WS\_VISIBLE or WS\_CHILD,

ExitButtonXPos, ExitButtonYPos, ExitButtonWidth, ExitButtonHeight,

hWnd, ExitButtonID, hInstance, NULL

; If the event is the window's command

; (button click, whatever)

.elseif uMsg==WM\_COMMAND

; If clicked "Exit", exit

cmp wParam, ExitButtonID

je stopProgram

; If clicked "Try", start checking the password

cmp wParam, ButtonID

jne stop

; Take out the string inside edit field

invoke SendMessage, hEditText, WM\_GETTEXT,

KeyboardHookWParam, addr input

; If the length of input and password aren't equal,

; there is an error

cmp ax, len

jne inputError

; Otherwise, compare input with password

cipherText input

compareText input, password

; If everything is correct -- zero (Z) flag is set to 0

; Otherwise (are not equal) -- Z flag is is set to 1

je inputSuccess

inputError:

printLabel ErrorMessageOne, nameTextXPos, nameTextYPos

printLabel ErrorMessageTwo, numberTextXPos, numberTextYPos

printLabel ErrorMessageThree, dateTextXPos, dateTextYPos

jmp stop

; Success dialog

inputSuccess:

printLabel nameText, nameTextXPos, nameTextYPos

printLabel numberText, numberTextXPos, numberTextYPos

printLabel dateText, dateTextXPos, dateTextYPos

.elseif uMsg == WM\_DESTROY

stopProgram:

invoke PostQuitMessage, 0

jmp stop

.else

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.endif

; Leave the procedure

stop:

xor eax,eax

ret

WndProc endp

end start

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Include-файл**

; ---------------------------------------------------------------------------------------------------

; 4-27-IP-93-Uzunlu.inc Copyright The MASM32 SDK 1998-2010

; ---------------------------------------------------------------------------------------------------

IFNDEF REKECHYNSKY\_INC

REKECHYNSKY\_INC equ <1>

printLabel macro printedText, xPos:=<LabelXPos>, yPos:=<LabelYPos>

; Create label using passed parameters

;; This comment won't be included to listing file

invoke CreateWindowEx, NULL,

addr LabelField, addr printedText,

WS\_VISIBLE or WS\_CHILD,

xPos, yPos, LabelWidth, LabelHeight,

hWnd, LabelID, hInstance, NULL

endm

cipherText macro ciphered

;; Declare local labels

local cycle

local endPoint

; Clear di register (it can be full with a value)

mov di, 0

cycle:

cmp di, len

je endPoint

mov dh, key

xor ciphered[di], dh

inc di

jne cycle

endPoint:

endm

compareText macro ciphered, passwordText

local cycle

local endPoint

mov di, 0

cycle:

cmp di, len

je endPoint

; If we didn't reach the end, check

mov dh, ciphered[di]

cmp dh, passwordText[di]

; If the letters aren't equal, end of comparing

jne endPoint

inc di

jmp cycle

;; Comment test

endPoint:

endm

ELSE

echo ---------------------------------------------------------

echo WARNING duplicate include file 4-18-IP-94-Rekechynsky.inc

echo ---------------------------------------------------------

ENDIF

Можна помітити директиви, які перевіряють наявність деякої константи REKECHYNSKY\_INC. Це дозволяє забезпечити включання цього файлу лише раз, і не більше. Корисно лише у тому разі, коли кілька імпортованих бібліотек включають у себе цей файл.

Тепер заглянемо у лістинг:

include 4-27-IP-93-Uzunlu.inc

C ; ---------------------------------------------------------------------------------------------------

C ; 4-27-IP-93-Uzunlu.inc Copyright The MASM32 SDK 1998-2010

C ; ---------------------------------------------------------------------------------------------------

C

C IFNDEF REKECHYNSKY\_INC

= 1 C REKECHYNSKY\_INC equ <1>

C

C printLabel macro printedText, xPos:=<LabelXPos>, yPos:=<LabelYPos>

C ; Create label using passed parameters

; This comment won't be included to listing file

C invoke CreateWindowEx, NULL,

C addr LabelField, addr printedText,

C WS\_VISIBLE or WS\_CHILD,

C xPos, yPos, LabelWidth, LabelHeight,

C hWnd, LabelID, hInstance, NULL

C endm

C

C cipherText macro ciphered

C

; Declare local labels

C local cycle

local endPoint

; Clear di register (it can be full with a value)

mov di, 0

cycle:

C cmp di, len

C je endPoint

C mov dh, key

C xor ciphered[di], dh

C inc di

jne cycle

C endPoint:

C endm

C

C compareText macro ciphered, passwordText

C local cycle

local endPoint

mov di, 0

cycle:

C cmp di, len

C je endPoint

C ; If we didn't reach the end, check

mov dh, ciphered[di]

C cmp dh, passwordText[di]

C ; If the letters aren't equal, end of comparing

jne endPoint

C inc di

jmp cycle

C

; Comment test

C endPoint:

C endm

C

C ELSE

C echo ---------------------------------------------------------

C echo WARNING duplicate include file 4-27-IP-93-Uzunlu.inc

C echo ---------------------------------------------------------

C ENDIF

C

C

Крім того, що ті коментарі так і не зникли, крім цього з’явились ще таємничі літери С.

Якщо полистати ще вище, то ця літера є перед кожним рядком Include-файлу.

Таким чином, ця літера позначає рядки з Include-файлів.

ВИСНОВОК. Розроблено програму із графічним інтерфейсом, яка запитує пароль. Перевірено на коректність. Досліджено лістинги файлів та зроблено висновки щодо роботи компоновщика. Роботу виконано успішно, я молодець.