НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Системне програмування »

*Звiт з лабораторної роботи №7*

*Тема: «*Обробка повідомлень клавіатури та миші *»*

Виконав:

ст. гр. KIT-119D

Гряник Г.В.

Перевірив:

Межерицький С.Г.

Харків – 2020

**Мета :**

Набути практичних навичок обробки повідомлень від клавіатури, написаних на мові асемблера з використанням API-функцій..

***Індивідуальне завдання***

Варіант 6.

* Використовувати функцію SetKeyboardState копіювання 256-байтового масиву станів клавіш клавіатури в таблицю станів введення з клавіатури.
* Створити два вікна, одне з яких - батьківське для іншого. Переміщення миші в некліентсой області дочірнього вікна призводить до його згортання. Натискання на клавішу «пробіл» при установці фокусу введення на батьківське вікно призводить також до його згортання.

**Текст програми**

Програма 1: (LR7-1)

include \masm64\include64\masm64rt.inc

.data

hInstance dq ? ; дескриптор програми

hWnd dq ? ; дескриптор окна

hIcon dq ? ; дескриптор иконки

hCursor dq ? ; дескриптор курсора

sWid dq ? ; ширина монитора (колич. пикселей по x)

sHgt dq ? ; высота монитора (колич. пикселей по y)

title1 db "Лаб.7 SetKeyboardStatemasm64",0

classname db "template\_class",0

caption db "SetKeyboardState",0

Hello db "Натисніть на символ. ",0

txt01 db "Використовувати функцію SetKeyboardState копіювання 256-байтового масиву станів клавіш клавіатури в таблицю станів введення з клавіатури.",0

autor db "Автор: Гряник Г.В., гр.КІТ-119Д",0

char1 dq 30h; символ пропуска, который программа получает от клавиатуры

x dq 0;

y dq 20;

Keys db 250 dup(81h ),0

K db 0;

.code

entry\_point proc

invoke GetKeyboardState , addr Keys

mov Keys[VK\_LSHIFT],1;

mov hInstance,rv(GetModuleHandle,0)

mov hIcon,rv(LoadImage,hInstance,10,IMAGE\_ICON,128,128,LR\_DEFAULTCOLOR)

; mov hBmp, rv(ResImageLoad,20,,128,128,LR\_DEFAULTCOLOR)

invoke DialogBoxParam,hInstance,1000,0,ADDR mainW,hIcon

invoke ExitProcess,0

ret

entry\_point endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

mainW proc hWin:QWORD,uMsg:QWORD,wParam:QWORD,lParam:QWORD

.switch uMsg

.case WM\_INITDIALOG ; сообщение о создании диал. окна

invoke SetWindowText,hWin,title1

invoke SendMessage,hWin,WM\_SETICON,1,lParam ; отправляет сообщение окну

invoke SendMessage,rv(GetDlgItem,hWin,101),\ ; сообщение окну по дескриптору органа управления

STM\_SETIMAGE,IMAGE\_ICON,lParam

; 102 - jpg

; mov hStatic, rv(GetDlgItem,hWin,101)

; invoke SendMessage,hStatic,STM\_SETIMAGE,IMAGE\_BITMAP,hBmp

.return TRUE

.case WM\_COMMAND ; сообщение от меню или кнопки

.switch wParam

.case 103 ; кнопка

invoke MsgboxI,hWin,ADDR txt01,"Завдання",MB\_OK,10

.case 105

invoke MsgboxI,hWin,ADDR autor,"Автор",MB\_OK,10

.case 106

mov hInstance,rv(GetModuleHandle,0) ; получение и сохранение дескрипторa програми

mov hIcon, rv(LoadIcon,hInstance,10) ; загрузка и сохранение дескрипторa иконки

mov hCursor,rv(LoadCursor,0,IDC\_ARROW) ; загрузка курсора и сохранение

mov sWid,rv(GetSystemMetrics,SM\_CXSCREEN) ; получение кол. пикселей по х монитора

mov sHgt,rv(GetSystemMetrics,SM\_CYSCREEN) ; получение кол. пикселей по y монитора

call main

.case 110 ; кнопка

invoke MsgboxI,hWin,ADDR txt01,"Завдання",MB\_OK,10

.case 113

invoke MsgboxI,hWin,ADDR autor,"Автор",MB\_OK,10

.case 112

mov hInstance,rv(GetModuleHandle,0) ; получение и сохранение дескрипторa програми

mov hIcon, rv(LoadIcon,hInstance,10) ; загрузка и сохранение дескрипторa иконки

mov hCursor,rv(LoadCursor,0,IDC\_ARROW) ; загрузка курсора и сохранение

mov sWid,rv(GetSystemMetrics,SM\_CXSCREEN) ; получение кол. пикселей по х монитора

mov sHgt,rv(GetSystemMetrics,SM\_CYSCREEN) ; получение кол. пикселей по y монитора

call main

.case 117

rcall SendMessage,hWin,WM\_SYSCOMMAND,SC\_CLOSE,0;

.case 115

rcall SendMessage,hWin,WM\_SYSCOMMAND,SC\_CLOSE,0;

.endsw

.case WM\_CLOSE ; если есть сообщение о закрытии окна

invoke EndDialog,hWin,0 ;

.endsw

xor rax, rax

ret

mainW endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main proc

LOCAL wc :WNDCLASSEX ; объявление локальных переменных

LOCAL lft :QWORD ; Лок. переменные содержатся в стеке

LOCAL top :QWORD ; и существуют только во время вып. проц.

LOCAL wid :QWORD

LOCAL hgt :QWORD

mov wc.cbSize,SIZEOF WNDCLASSEX ; колич. байтов структуры

mov wc.style,CS\_BYTEALIGNCLIENT or CS\_BYTEALIGNWINDOW ; стиль окна

mov wc.lpfnWndProc,ptr$(WndProc) ; адрес процедуры WndProc

mov wc.cbClsExtra,0 ; количество байтов для структуры класса

mov wc.cbWndExtra,0 ; количество байтов для структуры окна

mrm wc.hInstance,hInstance ; заполнение полЯ дескриптора в структуре

mrm wc.hIcon, hIcon ; хэндл иконки

mrm wc.hCursor,hCursor ; хэндл курсора

mrm wc.hbrBackground,0 ; цвет окна

mov wc.lpszMenuName,0 ; заполнение поля в структуре с именем ресурса меню

mov wc.lpszClassName,ptr$(classname) ; имя класса

mrm wc.hIconSm,hIcon

invoke RegisterClassEx,ADDR wc ; регистрациЯ класса окна

mov wid, 420 ; ширина пользовательского окна в пикселях

mov hgt, 620 ; высота пользовательского окна в пикселях

mov rax,sWid ; колич. пикселей монитора по x

sub rax,wid ; дельта • = •(монитора) - х(окна пользователя)

shr rax,1 ; получение середины •

mov lft,rax ;

mov rax, sHgt ; колич. пикселей монитора по y

sub rax, hgt ;

shr rax, 1 ;

mov top, rax ;

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

; centre window at calculated sizes

invoke CreateWindowEx,WS\_EX\_LEFT or WS\_EX\_ACCEPTFILES, \

ADDR classname,ADDR caption, \

WS\_OVERLAPPED or WS\_VISIBLE or WS\_SYSMENU,\

lft,top,wid,hgt,0,0,hInstance,0

mov hWnd,rax ; сохранение дескриптора окна

call msgloop

ret

main endp

msgloop proc

LOCAL msg :MSG

LOCAL pmsg :QWORD

mov pmsg,ptr$(msg) ; получение адреса структуры сообщениЯ

jmp gmsg ; jump directly to GetMessage()

mloop:

invoke TranslateMessage,pmsg

invoke DispatchMessage,pmsg ; отправка на обслуживание к WndProc

gmsg:

test rax, rv(GetMessage,pmsg,0,0,0) ; пока GetMessage не вернет ноль

jnz mloop

ret

msgloop endp

WndProc proc hWin:QWORD,uMsg:QWORD,wParam:QWORD,lParam:QWORD

LOCAL hdc:HDC ; резервирование стека для дескриптора окна

LOCAL ps:PAINTSTRUCT ; для структуры PAINTSTRUCT

LOCAL rect:RECT ; для структуры координат RECT

.switch uMsg

.case WM\_DESTROY ; если есть сообщение про уничтожение окна

invoke PostQuitMessage,NULL

.case WM\_CHAR ; если есть сообщение от клавиатуры

mov Keys[VK\_SHIFT],80h

mov Keys[WM\_CHAR+1],80h

invoke SetKeyboardState, addr Keys

push wParam ; то временно сохранить введенный символ

pop char1 ; и перезаписать в ячейку для выведа в окно

inc char1

invoke InvalidateRect,hWnd,0,TRUE ; вызов функции и WM\_PAINT

invoke GetKeyboardState, addr K

mov r15 ,x

cmp r15,420

JNE m2

mov x,0;

add y,20

m2: add x ,10

;invoke GetKeyboardState , addr Keys

;mov Keys[VK\_SHIFT],80h

; invoke SetKeyboardState, addr Keys

.case WM\_PAINT ; если есть смс о перерисовании

invoke BeginPaint,hWnd, ADDR ps ; вызов подготовительной процедуры

mov hdc,rax ; сохранение контекста

invoke TextOut,hdc,x,y,\ ; координаты начала текста

addr char1,1; адрес хранения текста и кол. байтов текста

invoke GetClientRect,hWnd,ADDR rect ; занесение в структуру rect

; характеристик окна

invoke DrawText, hdc,ADDR Hello,-1, ADDR rect, \ ; рисование текста

DT\_SINGLELINE or DT\_CENTER or DT\_VCENTER

invoke EndPaint,hWnd, ADDR ps ; завершение рисования

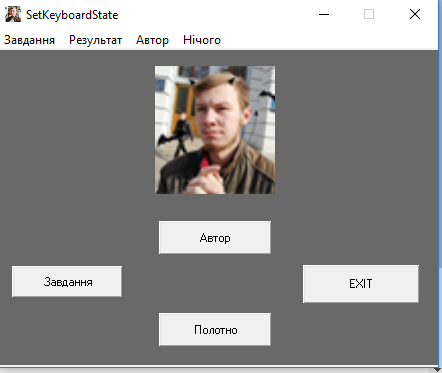
.endsw

invoke DefWindowProc,hWin,uMsg,wParam,lParam

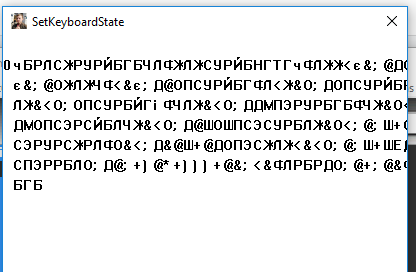
ret

WndProc endp

End

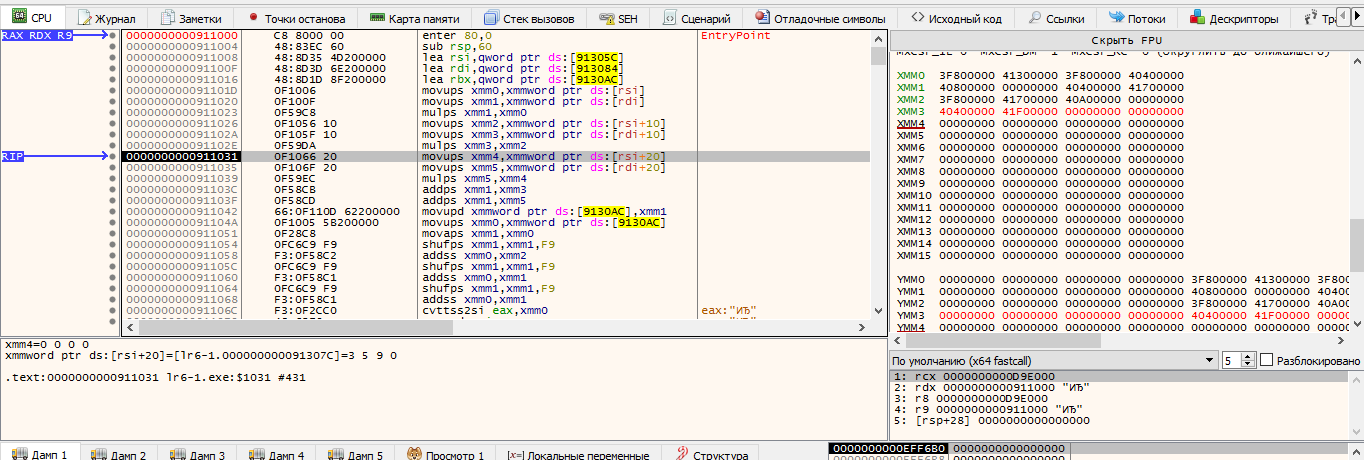


А)



Б)

**Рис 3 – результат виконання роботи програми** LR7-1



**Рис 4 – результат налагодження програми** LR7-1

Алгоритм: Програма імітує натискання деякої клавіші та додатково визначає яка клавіша натискається і реєструє не натиснуту клавішу, а наступну за списком.

**Програма 2**: (LR7-2)

include \masm64\include64\masm64rt.inc

.data

hInstance dq ? ; дескриптор програми

hWnd dq ? ; дескриптор окна

hIcon dq ? ; дескриптор иконки

hCursor dq ? ; дескриптор курсора

sWid dq ? ; ширина монитора (колич. пикселей по x)

sHgt dq ? ; высота монитора (колич. пикселей по y)

title1 db "Лаб.7 SetKeyboardStatemasm64",0

classname db "template\_class",0

caption db "Mouse",0

Hello db "Натисніть на символ. ",0

txt01 db "Переміщення миші в некліентсой області дочірнього вікна призводить до його згортання. Натискання на клавішу «пробіл» при установці фокусу введення на батьківське вікно призводить також до його згортання.",0

autor db "Автор: Гряник Г.В., гр.КІТ-119Д",0

.code

entry\_point proc

mov hInstance,rv(GetModuleHandle,0)

mov hIcon,rv(LoadImage,hInstance,10,IMAGE\_ICON,128,128,LR\_DEFAULTCOLOR)

; mov hBmp, rv(ResImageLoad,20,,128,128,LR\_DEFAULTCOLOR)

invoke DialogBoxParam,hInstance,1000,0,ADDR mainW,hIcon

invoke ExitProcess,0

ret

entry\_point endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

mainW proc hWin:QWORD,uMsg:QWORD,wParam:QWORD,lParam:QWORD

.switch uMsg

.case WM\_INITDIALOG ; сообщение о создании диал. окна

invoke SetWindowText,hWin,title1

invoke SendMessage,hWin,WM\_SETICON,1,lParam ; отправляет сообщение окну

invoke SendMessage,rv(GetDlgItem,hWin,101),\ ; сообщение окну по дескриптору органа управления

STM\_SETIMAGE,IMAGE\_ICON,lParam

; 102 - jpg

; mov hStatic, rv(GetDlgItem,hWin,101)

; invoke SendMessage,hStatic,STM\_SETIMAGE,IMAGE\_BITMAP,hBmp

.return TRUE

.case WM\_COMMAND ; сообщение от меню или кнопки

.switch wParam

.case 103 ; кнопка

invoke MsgboxI,hWin,ADDR txt01,"Завдання",MB\_OK,10

.case 105

invoke MsgboxI,hWin,ADDR autor,"Автор",MB\_OK,10

.case 106

mov hInstance,rv(GetModuleHandle,0) ; получение и сохранение дескрипторa програми

mov hIcon, rv(LoadIcon,hInstance,10) ; загрузка и сохранение дескрипторa иконки

mov hCursor,rv(LoadCursor,0,IDC\_ARROW) ; загрузка курсора и сохранение

mov sWid,rv(GetSystemMetrics,SM\_CXSCREEN) ; получение кол. пикселей по х монитора

mov sHgt,rv(GetSystemMetrics,SM\_CYSCREEN) ; получение кол. пикселей по y монитора

call main

.case 110 ; кнопка

invoke MsgboxI,hWin,ADDR txt01,"Завдання",MB\_OK,10

.case 113

invoke MsgboxI,hWin,ADDR autor,"Автор",MB\_OK,10

.case 112

mov hInstance,rv(GetModuleHandle,0) ; получение и сохранение дескрипторa програми

mov hIcon, rv(LoadIcon,hInstance,10) ; загрузка и сохранение дескрипторa иконки

mov hCursor,rv(LoadCursor,0,IDC\_ARROW) ; загрузка курсора и сохранение

mov sWid,rv(GetSystemMetrics,SM\_CXSCREEN) ; получение кол. пикселей по х монитора

mov sHgt,rv(GetSystemMetrics,SM\_CYSCREEN) ; получение кол. пикселей по y монитора

call main

.case 117

rcall SendMessage,hWin,WM\_SYSCOMMAND,SC\_CLOSE,0;

.case 115

rcall SendMessage,hWin,WM\_SYSCOMMAND,SC\_CLOSE,0;

.case WM\_DESTROY ; если есть сообщение про уничтожение окна

invoke PostQuitMessage,NULL

.case 107 ; если есть сообщение от клавиатуры

invoke keybd\_event,VK\_LWIN,0,0,NULL

invoke Sleep,1

invoke keybd\_event,'M',0,0,NULL

invoke Sleep,1

invoke keybd\_event,VK\_LWIN,0,KEYEVENTF\_KEYUP,NULL

.endsw

.case WM\_CLOSE ; если есть сообщение о закрытии окна

invoke EndDialog,hWin,0 ;

.endsw

xor rax, rax

ret

mainW endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main proc

LOCAL wc :WNDCLASSEX ; объявление локальных переменных

LOCAL lft :QWORD ; Лок. переменные содержатся в стеке

LOCAL top :QWORD ; и существуют только во время вып. проц.

LOCAL wid :QWORD

LOCAL hgt :QWORD

mov wc.cbSize,SIZEOF WNDCLASSEX ; колич. байтов структуры

mov wc.style,CS\_BYTEALIGNCLIENT or CS\_BYTEALIGNWINDOW ; стиль окна

mov wc.lpfnWndProc,ptr$(WndProc) ; адрес процедуры WndProc

mov wc.cbClsExtra,0 ; количество байтов для структуры класса

mov wc.cbWndExtra,0 ; количество байтов для структуры окна

mrm wc.hInstance,hInstance ; заполнение полЯ дескриптора в структуре

mrm wc.hIcon, hIcon ; хэндл иконки

mrm wc.hCursor,hCursor ; хэндл курсора

mrm wc.hbrBackground,5 ; цвет окна

mov wc.lpszMenuName,0 ; заполнение поля в структуре с именем ресурса меню

mov wc.lpszClassName,ptr$(classname) ; имя класса

mrm wc.hIconSm,hIcon

invoke RegisterClassEx,ADDR wc ; регистрациЯ класса окна

mov wid, 420 ; ширина пользовательского окна в пикселях

mov hgt, 420 ; высота пользовательского окна в пикселях

mov rax,sWid ; колич. пикселей монитора по x

sub rax,wid ; дельта • = •(монитора) - х(окна пользователя)

shr rax,1 ; получение середины •

mov lft,rax ;

mov rax, sHgt ; колич. пикселей монитора по y

sub rax, hgt ;

shr rax, 1 ;

mov top, rax ;

invoke CreateWindowEx,WS\_EX\_LEFT or WS\_EX\_ACCEPTFILES, \

ADDR classname,ADDR caption, \

WS\_OVERLAPPED or WS\_VISIBLE or WS\_OVERLAPPEDWINDOW,\

lft,top,wid,hgt,0,0,hInstance,0

mov hWnd,rax ; сохранение дескриптора окна

call msgloop

ret

main endp

msgloop proc

LOCAL msg :MSG

LOCAL pmsg :QWORD

mov pmsg,ptr$(msg) ; получение адреса структуры сообщениЯ

jmp gmsg ; jump directly to GetMessage()

mloop:

invoke TranslateMessage,pmsg

invoke DispatchMessage,pmsg ; отправка на обслуживание к WndProc

gmsg:

test rax, rv(GetMessage,pmsg,0,0,0) ; пока GetMessage не вернет ноль

jnz mloop

ret

msgloop endp

WndProc proc hWin:QWORD,uMsg:QWORD,wParam:QWORD,lParam:QWORD

LOCAL hdc:HDC ; резервирование стека для дескриптора окна

LOCAL ps:PAINTSTRUCT ; для структуры PAINTSTRUCT

LOCAL rect:RECT ; для структуры координат RECT

.switch uMsg

.case WM\_MOUSEMOVE ;сообщение на перевод окна

invoke keybd\_event,VK\_LWIN,0,0,NULL

invoke Sleep,1

invoke keybd\_event,'M',0,0,NULL

invoke Sleep,1

invoke keybd\_event,VK\_LWIN,0,KEYEVENTF\_KEYUP,NULL

.endif

invoke EndPaint,hWnd, ADDR ps

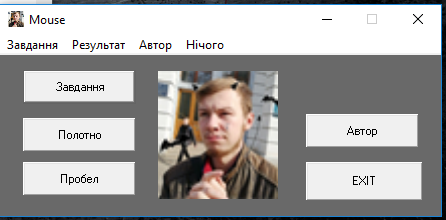
invoke DefWindowProc,hWin,uMsg,wParam,lParam

ret

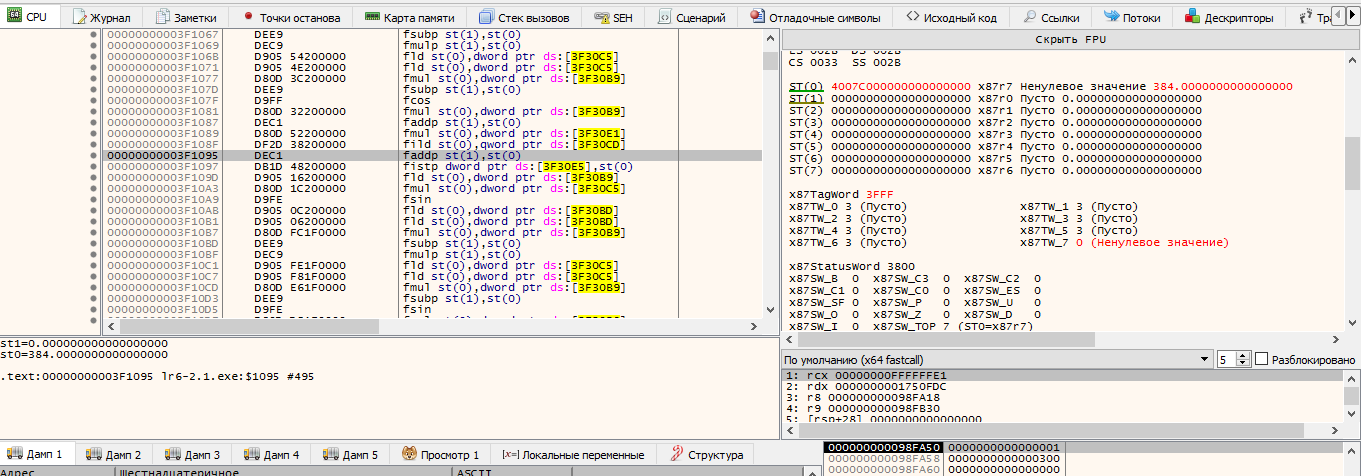
WndProc endp

end

**Результат**



**Рис 3 – результат виконання роботи програми** LR7-2



**Рис 4 – результат налагодження програми** LR7-2

**Алгоритм:** відкривається вікно , завдяки якому можна створити нове вікно , і якщо на нове вікно навести курсором, то ця дія призводить до згортання вікон.

**Висновок:** Під час лабораторної роботи було набуто практичних навичок з роботи mas64, а саме , навичок обробки повідомлень від клавіатури, написаних на мові асемблера з використанням API-функцій..