**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 №8**

з дисципліни

«Системне програмування»

на тему

«Розробка і використання динамічних бібліотек»

Виконав:

Перевірив:

студент групи ІП-93

Павлов Валерій Георгійович

Домінський Валентин Олексійович

номер залікової книжки: 9311

номер у списку: 9

Київ 2021

**Мета:**

Вивчення прийомів розробки і використання процедур, представлених у вигляді динамічних бібліотек

**Порядок виконання роботи:**

1. Вивчити методи розробки і використання DLL- бібліотек
2. Для свого варіанту індивідуального завдання до лабораторної роботи 6 розробити процедуру на мові Асемблер, в якій реалізувати обчислення заданого арифметичного виразу\* і виведення результатів у віконному інтерфейсі. Параметрами, які передаються до процедурі, є значення змінних. Процедуру розробити в двох варіантах:
   1. окремо процедура без вказівки точки входу;
   2. окремо процедура з використанням точки входу.
3. Розробити файл установок модуля DEF і виконати компіляцію обох варіантів файлу бібліотек DLL. Лістинги обох варіантів написання процедур, файлу DEF і командних рядків компіляції представити в звіті по лабораторній роботі.
4. Написати і скомпілювати файл основної програми, в якому передбачити неявне завантаження розробленої DLL- бібліотеки з передачею в неї в циклі 5 значень змінних\*\*, заданих дійсними числами у вигляді одновимірних масивів.
5. Перевірити роботу основної програми з підключенням по черзі спочатку одної, а потім іншої DLL- бібліотек і привести в звіті по лабораторній роботі лістинг основної програми і скріншоти вікон з результатами обчислень для обох варіантів.
6. Написати і скомпілювати файл основної програми, в якому передбачити явне завантаження розробленої DLL- бібліотеки з передачею до неї в циклі тих же 5 значень змінних.
7. Переконатися в працездатності основної програми при зверненні в ній по черзі спочатку одної, а потім іншої з DLL- бібліотек, розроблених в п. 3. Лістинг основної програми та скрішоти вікон з результатами обчислень за обома варіантами привести в звіті по лабораторній роботі.
8. Для перевірки правильності виконання розрахунків і результатів, що виводяться, заздалегідь виконати контрольні розрахунки. Проміжні і остаточні результати контрольних розрахунків привести в звіті по лабораторній роботі\*\*\*.
9. Виконати відладку програми шляхом порівняння розрахованих програмою результатів з контрольними прикладами.
10. При підключенні у програмі зовнішніх бібліотек та макросів додати до нього коментар, де вказати, які функції з них потрібні для використання у Вашій програмі.
11. Зробити висновки по лабораторній роботі

**Хід роботи**

1-7.

Сирцеві коди:

Динамічне підключення з точкою входу:

8-9-IP93-Dominskyi-Dynamic-Entry.asm:

; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

WinWarningProto proto **:dword,:dword,:dword**

WinMainProto proto **:dword,:dword,:dword**

; Libraries And Macroses

include **/**masm32**/**include**/**windows.inc ;; hInstance, WNDCLASSEX, SW\_SHOWDEFAULT, HWND && NULL

;; CreateWindowEx, DestroyWindow, PostQuitMessage, DefWindowProc, DispatchMessage && wsprintf

include **/**masm32**/**include**/**user32.inc

includelib **/**masm32**/**lib**/**user32.lib

;; GetModuleHandle, ExitProcess, LoadLibrary, GetProcAddress && FreeLibrary

include **/**masm32**/**include**/**kernel32.inc

includelib **/**masm32**/**lib**/**kernel32.lib

; Our Macroses

; We place them here, 'cause it won't degrade the readability of the code

; Macros #1 for printing some text

PrintInformationInWindow macro heightPosition**,** infoToShow

; for example, this commentary is included into macroexpansion

;; but this - not

; just pass position of the text on vertical

; and text, that We want to show

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheText**,** offset infoToShow**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_TEXT **or** SS\_CENTER **or** BS\_VCENTER**,**

16**,** heightPosition**,** 890**,** 75**,**

hWnd**,** 7044**,** hInstance**,** NULL

endm

.data?

hInstance HINSTANCE **?** ; Handle of our program

hWndOfWarnWindow HWND **?** ; Handle of our warn window

hWndOfMainWindow HWND **?** ; Handle of our main window

; address for Library

hLib **dword** **?**

; address for function

DoArithmeticOperationsAddress **dword** **?**

; Data Segment

.data

StartingText DB "У наступному вікні Ви побачите 5 різних арифметичних виразів"**,** 13**,** 0

TempPlaceForText DB 256 DUP**(**0**)**

; Name Of Message Box

MsgBoxName DB "8-9-IP93-Dominskyi"**,** 0

; Name of the Library to use

LibraryName DB "8-9-IP93-Dominskyi-Dynamic-Entry-Library"**,** 0

; Name of the Function to use

DoArithmeticOperationsFunctionFromTheLib DB "DoArithmeticOperations"**,** 0

NameOfTheWarnWindows DB "Window with warn text"**,** 0 ; the name of our warn window class

NameOfMainWindows DB "Window with main text"**,** 0 ; the name of our success window class

NameOfTheButton DB "Button"**,** 0 ; the name of our button class

NameOfTheText DB "Static"**,** 0 ; the name of our text class

TextForOKButton DB "ОК"**,** 0

; My equation = (2 \* c - d / 23) / (ln( b - a / 4))

; can't be 1 or 0

; first way of declaring array

FloatsA dq 2.0**,** **-**16.0**,** **-**68.946**,** 0.001**,** 4.0 ;; first numbers

FloatsB dq 4.0**,** 23.091**,** 6.67**,** **-**3.33**,** 2.0 ;; second numbers

FloatsC dq **-**99.0**,** **-**2.111**,** **-**78.2**,** 123.4**,** 44.47 ;; third numbers

; and the second one

FloatsD dq **-**15.125 ;; fourth numbers

dq 0.5

dq **-**22.1

dq **-**9.0

dq 12.2222

; for automating

possibleHeight DD 25

coefficientOfMultiplyingForTextHeight DD 3

; first text to show

variantToShow DB "My equation = (2 \* c - d / 23) / (ln(b - a / 4))"**,** 13**,** 0

; Code Segment

.code

start**:** ; Generates program start-up code

invoke WinWarningProto**,** hInstance**,**NULL**,** SW\_SHOWDEFAULT ;invoke function

invoke GetModuleHandle**,** NULL

**mov** hInstance**,** **eax**

invoke WinMainProto**,** hInstance**,**NULL**,** SW\_SHOWDEFAULT ;invoke function

invoke ExitProcess**,** **eax** ; quit program. code returns in EAX register from Main Function.

; function declaration of WinWarn

WinWarningProto proc hInst**:**HINSTANCE**,**hPrevInst**:**HINSTANCE**,**CmdShow**:dword**

; there we need LOCAL variables

LOCAL wc**:**WNDCLASSEX

LOCAL msg**:**MSG

LOCAL hwnd**:**HWND

; assign variables of WNDCLASSEX

; window class is a specification of a window

**mov** wc.cbSize**,** sizeof WNDCLASSEX

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

**mov** wc.lpfnWndProc**,** offset WndWarnProc

**mov** wc.cbClsExtra**,** NULL

**mov** wc.cbWndExtra**,** NULL

**push** hInstance

**pop** wc.hInstance

**mov** wc.hbrBackground**,** COLOR\_WINDOW**+**1

**mov** wc.lpszMenuName**,** NULL

**mov** wc.lpszClassName**,** offset NameOfTheWarnWindows

invoke LoadIcon**,** NULL**,** IDI\_APPLICATION

**mov** wc.hIcon**,** **eax**

**mov** wc.hIconSm**,** **eax**

invoke LoadCursor**,** NULL**,** IDC\_ARROW

**mov** wc.hCursor**,** **eax**

; create class of the window

invoke RegisterClassEx**,** **addr** wc

invoke CreateWindowEx**,** NULL**,**

offset NameOfTheWarnWindows**,**

offset MsgBoxName**,**

WS\_OVERLAPPEDWINDOW **or** DS\_CENTER**,**

470**,** 310**,** 300**,** 150**,**

NULL**,** NULL**,** hInst**,** NULL

**mov** hWndOfWarnWindow**,** **eax**

; write window handle in eax

**mov** hwnd**,eax**

; Show window

invoke ShowWindow**,** hwnd**,**CmdShow

; update screen

invoke UpdateWindow**,** hwnd

; waits for message

.while TRUE

;returns FALSE IF WM\_QUIT message is received and will kill the loop

invoke GetMessage**,** **addr** msg**,**NULL**,**0**,**0

.break .IF **(**!**eax)**

;takes raw keyboard input and generates a new message

invoke TranslateMessage**,** **addr** msg

;sends the message data to the window procedure responsible for the specific window the message is for

invoke DispatchMessage**,** **addr** msg

; end while

.endw

; code returns in EAX register from Main Function.

**mov** **eax,** msg.wParam

; return

**ret**

;The ENDP directive defines the end of the procedure

;and has the same name as in the PROC directive

WinWarningProto endp

; function declaration of WinSuccess

WinMainProto proc hInst**:**HINSTANCE**,**hPrevInst**:**HINSTANCE**,**CmdShow**:dword**

; there we need LOCAL variables

LOCAL wc**:**WNDCLASSEX

LOCAL msg**:**MSG

LOCAL hwnd**:**HWND

; assign variables of WNDCLASSEX

; window class is a specification of a window

**mov** wc.cbSize**,** sizeof WNDCLASSEX

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

**mov** wc.lpfnWndProc**,** offset WndMainProc

**mov** wc.cbClsExtra**,** NULL

**mov** wc.cbWndExtra**,** NULL

**push** hInstance

**pop** wc.hInstance

**mov** wc.hbrBackground**,** COLOR\_WINDOW**+**1

**mov** wc.lpszMenuName**,** NULL

**mov** wc.lpszClassName**,** offset NameOfMainWindows

invoke LoadIcon**,** NULL**,** IDI\_APPLICATION

**mov** wc.hIcon**,** **eax**

**mov** wc.hIconSm**,** **eax**

invoke LoadCursor**,** NULL**,** IDC\_ARROW

**mov** wc.hCursor**,** **eax**

; create class of the window

invoke RegisterClassEx**,** **addr** wc

invoke CreateWindowEx**,** NULL**,**

offset NameOfMainWindows**,**

offset MsgBoxName**,**

WS\_OVERLAPPEDWINDOW **or** DS\_CENTER**,**

170**,** 50**,** 940**,** 550**,**

NULL**,** NULL**,** hInst**,** NULL

**mov** hWndOfMainWindow**,** **eax**

; write window handle in eax

**mov** hwnd**,eax**

; Show window

invoke ShowWindow**,** hwnd**,**CmdShow

; update screen

invoke UpdateWindow**,** hwnd

; waits for message

.while TRUE

;returns FALSE IF WM\_QUIT message is received and will kill the loop

invoke GetMessage**,** **addr** msg**,**NULL**,**0**,**0

.break .IF **(**!**eax)**

;takes raw keyboard input and generates a new message

invoke TranslateMessage**,** **addr** msg

;sends the message data to the window procedure responsible for the specific window the message is for

invoke DispatchMessage**,** **addr** msg

; end while

.endw

; code returns in EAX register from Main Function.

**mov** **eax,** msg.wParam

; return

**ret**

;The ENDP directive defines the end of the procedure

;and has the same name as in the PROC directive

WinMainProto endp

WndMainProc proc hWnd**:**HWND**,** ourMSG**:**UINT**,** wParam**:**WPARAM**,** lParam**:**LPARAM

; on window close

.IF ourMSG**==**WM\_CLOSE

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSEIF ourMSG**==**WM\_CREATE

**mov** **edi,** 0

; invoke macros #1 one time to create text

PrintInformationInWindow possibleHeight**,** offset variantToShow

; Dynamically Load Library

invoke LoadLibrary**,** **addr** LibraryName

; save descriptor

**mov** hLib**,** **eax**

; get function

invoke GetProcAddress**,** hLib**,** **addr** DoArithmeticOperationsFunctionFromTheLib

; save function

**mov** DoArithmeticOperationsAddress**,** **eax**

;; do the loop

LoopItself**:**

**lea** **ebx,** TempPlaceForText

**push** **ebx**

**lea** **ebx,** FloatsD**[**8**\*edi]**

**push** **ebx**

**lea** **ebx,** FloatsC**[**8**\*edi]**

**push** **ebx**

**lea** **ebx,** FloatsB**[**8**\*edi]**

**push** **ebx**

**lea** **ebx,** FloatsA**[**8**\*edi]**

**push** **ebx**

**call** **[**DoArithmeticOperationsAddress**]**

; mov possibleHeight into eax

**mov** **eax,** possibleHeight

;; Convert byte to word

**cbw**

; mov possibleHeight into ebx

**mov** **ebx,** coefficientOfMultiplyingForTextHeight

;; Convert byte to word

**cbw**

;; coefficientOfMultiplyingForTextHeight \* possibleHeight

;; eax \* ebx

**imul** **ebx**

**imul** **esi**

; print text

PrintInformationInWindow **eax,** TempPlaceForText

**inc** **edi**

**inc** **esi**

**cmp** **edi,** 5

**jne** LoopItself

; free library

invoke FreeLibrary**,** hLib

; create button

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheButton**,** offset TextForOKButton**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_CENTER **or** BS\_TEXT **or** BS\_VCENTER**,**

395**,** 465**,** 150**,** 30**,**

hWnd**,** 7033**,** hInstance**,** NULL

.ELSEIF ourMSG**==**WM\_COMMAND

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSE

; process the message

invoke DefWindowProc**,**hWnd**,**ourMSG**,**wParam**,**lParam

**ret**

.ENDIF

ExitCode**:**

**xor** **eax,** **eax**

**ret**

WndMainProc endp

WndWarnProc proc hWnd**:**HWND**,** ourMSG**:**UINT**,** wParam**:**WPARAM**,** lParam**:**LPARAM

; on window close

.IF ourMSG**==**WM\_CLOSE

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSEIF ourMSG**==**WM\_CREATE

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheButton**,** offset TextForOKButton**,**

WS\_CHILD **or** WS\_VISIBLE **or** BS\_CENTER **or** BS\_TEXT **or** BS\_VCENTER**,**

65**,** 65**,** 150**,** 30**,**

hWnd**,** 7003**,** hInstance**,** NULL

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheText**,** offset StartingText**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_TEXT **or** SS\_CENTER **or** BS\_VCENTER**,**

16**,** 10**,** 250**,** 50**,**

hWnd**,** 7004**,** hInstance**,** NULL

.ELSEIF ourMSG**==**WM\_COMMAND

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSE

; process the message

invoke DefWindowProc**,**hWnd**,**ourMSG**,**wParam**,**lParam

**ret**

.ENDIF

ExitCode**:**

**xor** **eax,eax**

**ret**

WndWarnProc endp

end start

8-9-IP93-Dominskyi-Dynamic-Entry-Library.asm:

; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

; Libraries And Macroses

;; implementation of FpuFLtoA

includelib **/**masm32**/**lib**/**Fpu.lib

include **/**masm32**/**include**/**Fpu.inc

;; CreateWindowEx, DestroyWindow, PostQuitMessage, DefWindowProc, DispatchMessage && wsprintf

include **/**masm32**/**include**/**user32.inc

includelib **/**masm32**/**lib**/**user32.lib

; ;; implementation of FloatToStr2

include **/**masm32**/**include**/**masm32.inc

includelib **/**masm32**/**lib**/**masm32.lib

.data?

; Start = (2 \* c - d / 23) / (ln(b - a / 4))

; Buffers for final float numbers

BufferFloatA DB 32 DUP**(?)**

BufferFloatB DB 32 DUP**(?)**

BufferFloatC DB 32 DUP**(?)**

BufferFloatD DB 32 DUP**(?)**

BufferFloatFinal DB 32 DUP**(?)**

; Buffers for intermediate results

; First Step

; Value of 2 \* c

BufferTwoMulC DB 32 DUP**(?)**

; Value of d / 23

BufferDdivTwenThree DB 32 DUP**(?)**

; Value of a - 4

BufferAdivFour DB 32 DUP**(?)**

; Second Step

; Value of 2 \* c - d / 23

BufferFirstPart DB 32 DUP**(?)**

; Value of b - a / 4

BufferBsubPartOfLn DB 32 DUP**(?)**

; Third Step

; Value of ln(b - a / 4)

BufferSecondPart DB 32 DUP**(?)**

; Data Segment

.data

ZeroDivisionText DB "Даний вираз має ділення на нуль. Перевірте Свої значення"**,** 13**,** 0

NegativeOrZeroLnText DB "Даний вираз має негативне число або нуль в (ln). Перевірте Свої значення"**,** 13**,** 0

; Name Of Message Box

MsgBoxName DB "8-9-IP93-Dominskyi"**,** 0

firstConstant dq 2.0

secondConstant dq 23.0

thirdConstant dq 4.0

zero dq 0.0

negativeZero dq **-**0.0

;; global variables for interpolating for main window

;; (I will put some int into them and show in main window)

;; mostly used for negative nums

floatFinal DQ 0

; form, which I will be filling with variables

equationVariables DB "For a = (%s), b = (%s), c = (%s) and d = (%s) We have (2 \* (%s) - (%s) / 23) / (ln((%s) - (%s) / 4)) = ((%s) - (%s)) / (ln((%s) - (%s))) = (%s) / (ln((%s))) = (%s) / (%s) = (%s)"**,** 13**,** 0

; Code Segment

.code

DLLmain proc hInstDLL**:** **DWORD,** reason**:** **DWORD,** unused**:** **DWORD**

**mov** **eax,** 1

**ret**

DLLmain endp

; procedure #1 for calculating

DoArithmeticOperations proc aFloat**:** ptr **qword,** bFloat**:** ptr **qword,** cFloat**:** ptr **qword,** dFloat**:** ptr **qword,** TempPlaceForText**:** **dword**

; My equation = (2 \* c - d / 23) / (ln(b - a / 4))

;; values for equation

**mov** **eax,** aFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatA

**mov** **eax,** bFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatB

**mov** **eax,** cFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatC

**mov** **eax,** dFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatD

**finit** ; FPU Initialization

; 2 \* c

; move 2 into st(0)

**fld** firstConstant

; move c into st(0) and 2 into st(1)

**mov** **eax,** cFloat

**fld** **qword** ptr **[eax]**

; multiply 2 by c and move result into st(0)

**fmul**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferTwoMulC**,** SRC1\_FPU **or** SRC2\_DIMM

; d / 23

; move d into st(0) and 2\*c into st(1)

**mov** **eax,** dFloat

**fld** **qword** ptr **[eax]**

; move 23 into st(0), d into st(1) and 2\*c into st(2)

**fld** secondConstant

; divide d by 23 and move result into st(0), 2\*c to st(1)

**fdiv**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferDdivTwenThree**,** SRC1\_FPU **or** SRC2\_DIMM

; 2 \* c - d / 23

; subtract d/23 from 2\*c, move result into st(0)

**fsub**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferFirstPart**,** SRC1\_FPU **or** SRC2\_DIMM

; move ln(2) into st(0) and 2\*c-d/23 into st(1)

**fldln2**

; move b into st(0), ln(2) into st(1) and 2\*c-d/23 into st(2)

**mov** **eax,** bFloat

**fld** **qword** ptr **[eax]**

; move a into st(0), b into st(1), ln(2) into st(2) 2\*c-d/23 into st(3)

**mov** **eax,** aFloat

**fld** **qword** ptr **[eax]**

; move 4 into st(0), a into st(1), b into st(2), ln(2) into st(3) and 2\*c-d/23 into st(4)

**fld** thirdConstant

; divide a by 4 and move it into st(0), b into st(1), ln(2) into st(2) and 2\*c-d/23 into st(3)

**fdiv**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferAdivFour**,** SRC1\_FPU **or** SRC2\_DIMM

; subtract a/4 from b, move result into st(0), ln(2) into st(1) and 2\*c-d/23 into st(2)

**fsub**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferBsubPartOfLn**,** SRC1\_FPU **or** SRC2\_DIMM

; compare, if number is zero or less for ln

; compares the contents of st (0) to the source

**fcom** zero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero

**je** NumberIsLessOrZero

; jump, if less than zero

**jb** NumberIsLessOrZero

; find ln(b - a/4) and move it into st(0), 2\*c-d/23 into st(1)

**fyl2x**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferSecondPart**,** SRC1\_FPU **or** SRC2\_DIMM

; compare, if number is zero for dividing

; compares the contents of st (0) to the source

**fcom** zero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero.zero

**je** NumberIsZero

; compares the contents of st (0) to the source

**fcom** negativeZero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to negative zero.zero

**je** NumberIsZero

; compares the contents of st (0) to zero

**ftst**

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero

**je** NumberIsZero

; divides 2\*c-d/23 by ln(b-a/4) and move it into st(0)

**fdiv**

; (2 \* c - d / 23) / (ln( b - a / 4))

; saves st(0) into variable

**fstp** floatFinal

;; value for final result

invoke FloatToStr2**,** floatFinal**,** **addr** BufferFloatFinal

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** equationVariables**,**

**addr** BufferFloatA**,** **addr** BufferFloatB**,** **addr** BufferFloatC**,** **addr** BufferFloatD**,**

**addr** BufferFloatC**,** **addr** BufferFloatD**,** **addr** BufferFloatB**,** **addr** BufferFloatA**,**

**addr** BufferTwoMulC**,** **addr** BufferDdivTwenThree**,** **addr** BufferFloatB**,**

**addr** BufferAdivFour**,** **addr** BufferFirstPart**,** **addr** BufferBsubPartOfLn**,**

**addr** BufferFirstPart**,** **addr** BufferSecondPart**,** **addr** BufferFloatFinal

**jmp** EndThisMacros

NumberIsZero**:**

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** ZeroDivisionText

**jmp** EndThisMacros

NumberIsLessOrZero**:**

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** NegativeOrZeroLnText

**jmp** EndThisMacros

EndThisMacros**:**

**ret**

DoArithmeticOperations endp

end DLLmain

8-9-IP93-Dominskyi-Dynamic-Entry.bat:

:: Our variables (two .asm files, two .obj files, dll and name of the .exe)

**set** NameOfTheFileAsASMParametr**=**"8-9-IP93-Dominskyi-Dynamic-Entry.asm"

**set** NameOfTheFileAsOBJParametr**=**"8-9-IP93-Dominskyi-Dynamic-Entry.obj"

**set** NameOfTheFileAsEXEParametr**=**"8-9-IP93-Dominskyi-Dynamic-Entry.exe"

**set** NameOfTheLibraryAsASMParametr**=**"8-9-IP93-Dominskyi-Dynamic-Entry-Library.asm"

**set** NameOfTheLibraryAsDLLParametr**=**"8-9-IP93-Dominskyi-Dynamic-Entry-Library.dll"

**set** NameOfTheLibraryAsOBJParametr**=**"8-9-IP93-Dominskyi-Dynamic-Entry-Library.obj"

:: There We are combining main file with dll one .exe

:: We can write there, for example, %OurDisk%\masm32\bin\ml, but We have masm commands in environment variables, so need to write only relative path

\masm32\bin\ml /c /coff "**%NameOfTheLibraryAsASMParametr%**"

\masm32\bin\Link.exe /out:"**%NameOfTheLibraryAsDLLParametr%**" /dll /export:DoArithmeticOperations "**%NameOfTheLibraryAsOBJParametr%**"

\masm32\bin\ml /c /coff "**%NameOfTheFileAsASMParametr%**"

\masm32\bin\Link.exe /subsystem:console "**%NameOfTheFileAsOBJParametr%**"

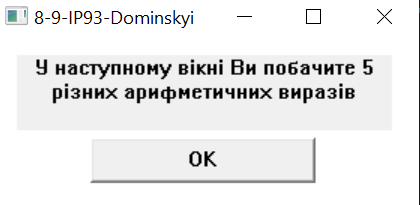
:: if You want window to pause after procedure, then uncomment next row (but You need it, only if You have some problem with code)

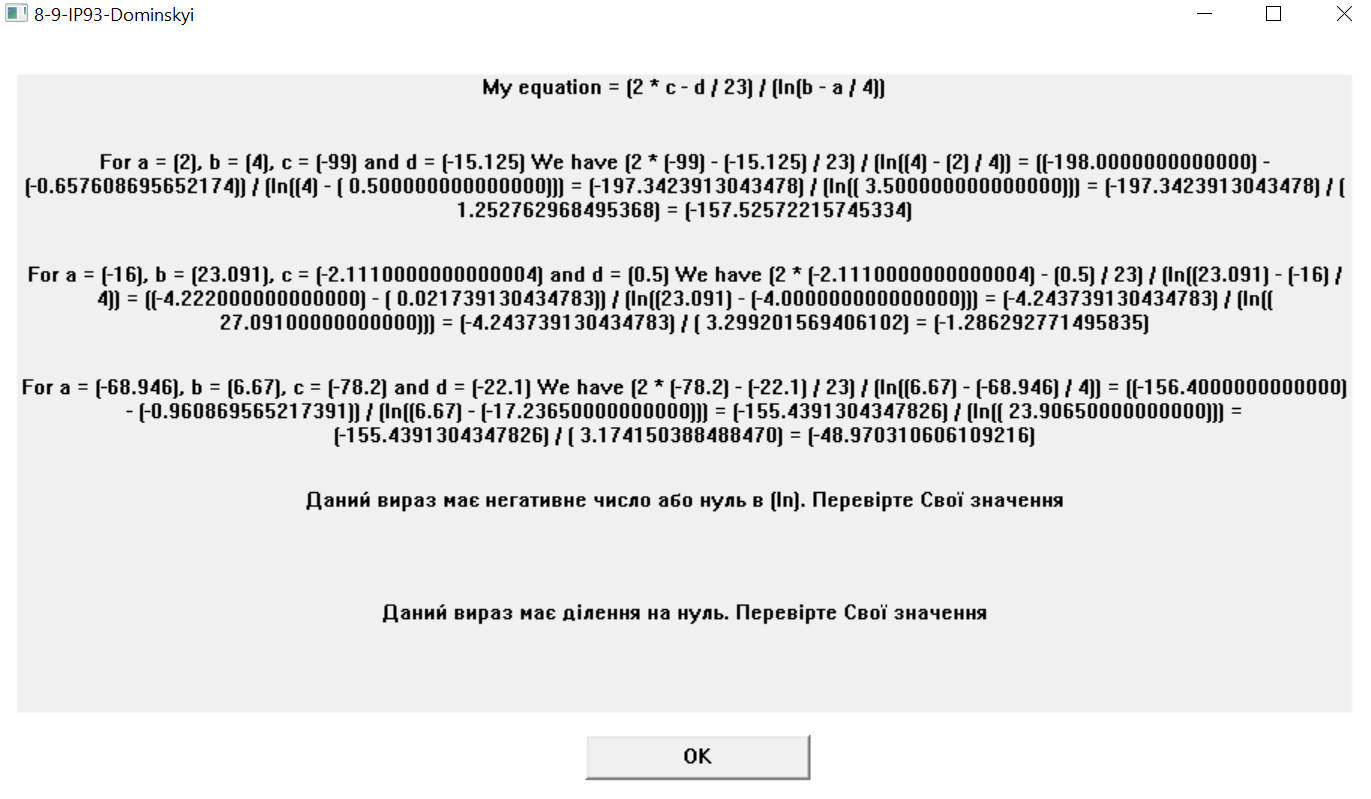
:: pause

:: if You don't want program to run after procedure, then comment next row

**%NameOfTheFileAsEXEParametr%**

Скріншоти:





Динамічне підключення без точки входу:

8-9-IP93-Dominskyi-Dynamic-WithoutEntry.asm:

; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

WinWarningProto proto **:dword,:dword,:dword**

WinMainProto proto **:dword,:dword,:dword**

; Libraries And Macroses

include **/**masm32**/**include**/**windows.inc ;; hInstance, WNDCLASSEX, SW\_SHOWDEFAULT, HWND && NULL

;; CreateWindowEx, DestroyWindow, PostQuitMessage, DefWindowProc, DispatchMessage && wsprintf

include **/**masm32**/**include**/**user32.inc

includelib **/**masm32**/**lib**/**user32.lib

;; GetModuleHandle, ExitProcess, LoadLibrary, GetProcAddress && FreeLibrary

include **/**masm32**/**include**/**kernel32.inc

includelib **/**masm32**/**lib**/**kernel32.lib

; Our Macroses

; We place them here, 'cause it won't degrade the readability of the code

; Macros #1 for printing some text

PrintInformationInWindow macro heightPosition**,** infoToShow

; for example, this commentary is included into macroexpansion

;; but this - not

; just pass position of the text on vertical

; and text, that We want to show

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheText**,** offset infoToShow**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_TEXT **or** SS\_CENTER **or** BS\_VCENTER**,**

16**,** heightPosition**,** 890**,** 75**,**

hWnd**,** 7044**,** hInstance**,** NULL

endm

.data?

hInstance HINSTANCE **?** ; Handle of our program

hWndOfWarnWindow HWND **?** ; Handle of our warn window

hWndOfMainWindow HWND **?** ; Handle of our main window

; address for Library

hLib **dword** **?**

; address for function

DoArithmeticOperationsAddress **dword** **?**

; Data Segment

.data

StartingText DB "У наступному вікні Ви побачите 5 різних арифметичних виразів"**,** 13**,** 0

TempPlaceForText DB 256 DUP**(**0**)**

; Name Of Message Box

MsgBoxName DB "8-9-IP93-Dominskyi"**,** 0

; Name of the Library to use

LibraryName DB "8-9-IP93-Dominskyi-Dynamic-WithoutEntry-Library"**,** 0

; Name of the Function to use

DoArithmeticOperationsFunctionFromTheLib DB "DoArithmeticOperations"**,** 0

NameOfTheWarnWindows DB "Window with warn text"**,** 0 ; the name of our warn window class

NameOfMainWindows DB "Window with main text"**,** 0 ; the name of our success window class

NameOfTheButton DB "Button"**,** 0 ; the name of our button class

NameOfTheText DB "Static"**,** 0 ; the name of our text class

TextForOKButton DB "OK"**,** 0

; My equation = (2 \* c - d / 23) / (ln( b - a / 4))

; can't be 1 or 0

; first way of declaring array

FloatsA dq 2.0**,** **-**16.0**,** **-**68.946**,** 0.001**,** 4.0 ;; first numbers

FloatsB dq 4.0**,** 23.091**,** 6.67**,** **-**3.33**,** 2.0 ;; second numbers

FloatsC dq **-**99.0**,** **-**2.111**,** **-**78.2**,** 123.4**,** 44.47 ;; third numbers

; and the second one

FloatsD dq **-**15.125 ;; fourth numbers

dq 0.5

dq **-**22.1

dq **-**9.0

dq 12.2222

; for automating

possibleHeight DD 25

coefficientOfMultiplyingForTextHeight DD 3

; first text to show

variantToShow DB "My equation = (2 \* c - d / 23) / (ln(b - a / 4))"**,** 13**,** 0

; Code Segment

.code

start**:** ; Generates program start-up code

invoke WinWarningProto**,** hInstance**,**NULL**,** SW\_SHOWDEFAULT ;invoke function

invoke GetModuleHandle**,** NULL

**mov** hInstance**,** **eax**

invoke WinMainProto**,** hInstance**,**NULL**,** SW\_SHOWDEFAULT ;invoke function

invoke ExitProcess**,** **eax** ; quit program. code returns in EAX register from Main Function.

; function declaration of WinWarn

WinWarningProto proc hInst**:**HINSTANCE**,**hPrevInst**:**HINSTANCE**,**CmdShow**:dword**

; there we need LOCAL variables

LOCAL wc**:**WNDCLASSEX

LOCAL msg**:**MSG

LOCAL hwnd**:**HWND

; assign variables of WNDCLASSEX

; window class is a specification of a window

**mov** wc.cbSize**,** sizeof WNDCLASSEX

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

**mov** wc.lpfnWndProc**,** offset WndWarnProc

**mov** wc.cbClsExtra**,** NULL

**mov** wc.cbWndExtra**,** NULL

**push** hInstance

**pop** wc.hInstance

**mov** wc.hbrBackground**,** COLOR\_WINDOW**+**1

**mov** wc.lpszMenuName**,** NULL

**mov** wc.lpszClassName**,** offset NameOfTheWarnWindows

invoke LoadIcon**,** NULL**,** IDI\_APPLICATION

**mov** wc.hIcon**,** **eax**

**mov** wc.hIconSm**,** **eax**

invoke LoadCursor**,** NULL**,** IDC\_ARROW

**mov** wc.hCursor**,** **eax**

; create class of the window

invoke RegisterClassEx**,** **addr** wc

invoke CreateWindowEx**,** NULL**,**

offset NameOfTheWarnWindows**,**

offset MsgBoxName**,**

WS\_OVERLAPPEDWINDOW **or** DS\_CENTER**,**

470**,** 310**,** 300**,** 150**,**

NULL**,** NULL**,** hInst**,** NULL

**mov** hWndOfWarnWindow**,** **eax**

; write window handle in eax

**mov** hwnd**,eax**

; Show window

invoke ShowWindow**,** hwnd**,**CmdShow

; update screen

invoke UpdateWindow**,** hwnd

; waits for message

.while TRUE

;returns FALSE IF WM\_QUIT message is received and will kill the loop

invoke GetMessage**,** **addr** msg**,**NULL**,**0**,**0

.break .IF **(**!**eax)**

;takes raw keyboard input and generates a new message

invoke TranslateMessage**,** **addr** msg

;sends the message data to the window procedure responsible for the specific window the message is for

invoke DispatchMessage**,** **addr** msg

; end while

.endw

; code returns in EAX register from Main Function.

**mov** **eax,** msg.wParam

; return

**ret**

;The ENDP directive defines the end of the procedure

;and has the same name as in the PROC directive

WinWarningProto endp

; function declaration of WinSuccess

WinMainProto proc hInst**:**HINSTANCE**,**hPrevInst**:**HINSTANCE**,**CmdShow**:dword**

; there we need LOCAL variables

LOCAL wc**:**WNDCLASSEX

LOCAL msg**:**MSG

LOCAL hwnd**:**HWND

; assign variables of WNDCLASSEX

; window class is a specification of a window

**mov** wc.cbSize**,** sizeof WNDCLASSEX

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

**mov** wc.lpfnWndProc**,** offset WndMainProc

**mov** wc.cbClsExtra**,** NULL

**mov** wc.cbWndExtra**,** NULL

**push** hInstance

**pop** wc.hInstance

**mov** wc.hbrBackground**,** COLOR\_WINDOW**+**1

**mov** wc.lpszMenuName**,** NULL

**mov** wc.lpszClassName**,** offset NameOfMainWindows

invoke LoadIcon**,** NULL**,** IDI\_APPLICATION

**mov** wc.hIcon**,** **eax**

**mov** wc.hIconSm**,** **eax**

invoke LoadCursor**,** NULL**,** IDC\_ARROW

**mov** wc.hCursor**,** **eax**

; create class of the window

invoke RegisterClassEx**,** **addr** wc

invoke CreateWindowEx**,** NULL**,**

offset NameOfMainWindows**,**

offset MsgBoxName**,**

WS\_OVERLAPPEDWINDOW **or** DS\_CENTER**,**

170**,** 50**,** 940**,** 550**,**

NULL**,** NULL**,** hInst**,** NULL

**mov** hWndOfMainWindow**,** **eax**

; write window handle in eax

**mov** hwnd**,eax**

; Show window

invoke ShowWindow**,** hwnd**,**CmdShow

; update screen

invoke UpdateWindow**,** hwnd

; waits for message

.while TRUE

;returns FALSE IF WM\_QUIT message is received and will kill the loop

invoke GetMessage**,** **addr** msg**,**NULL**,**0**,**0

.break .IF **(**!**eax)**

;takes raw keyboard input and generates a new message

invoke TranslateMessage**,** **addr** msg

;sends the message data to the window procedure responsible for the specific window the message is for

invoke DispatchMessage**,** **addr** msg

; end while

.endw

; code returns in EAX register from Main Function.

**mov** **eax,** msg.wParam

; return

**ret**

;The ENDP directive defines the end of the procedure

;and has the same name as in the PROC directive

WinMainProto endp

WndMainProc proc hWnd**:**HWND**,** ourMSG**:**UINT**,** wParam**:**WPARAM**,** lParam**:**LPARAM

; on window close

.IF ourMSG**==**WM\_CLOSE

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSEIF ourMSG**==**WM\_CREATE

**mov** **edi,** 0

; invoke macros #1 one time to create text

PrintInformationInWindow possibleHeight**,** offset variantToShow

; Dynamically Load Library

invoke LoadLibrary**,** **addr** LibraryName

; save descriptor

**mov** hLib**,** **eax**

; get function

invoke GetProcAddress**,** hLib**,** **addr** DoArithmeticOperationsFunctionFromTheLib

; save function

**mov** DoArithmeticOperationsAddress**,** **eax**

;; do the loop

LoopItself**:**

**lea** **ebx,** TempPlaceForText

**push** **ebx**

**lea** **ebx,** FloatsD**[**8**\*edi]**

**push** **ebx**

**lea** **ebx,** FloatsC**[**8**\*edi]**

**push** **ebx**

**lea** **ebx,** FloatsB**[**8**\*edi]**

**push** **ebx**

**lea** **ebx,** FloatsA**[**8**\*edi]**

**push** **ebx**

**call** **[**DoArithmeticOperationsAddress**]**

; mov possibleHeight into eax

**mov** **eax,** possibleHeight

;; Convert byte to word

**cbw**

; mov possibleHeight into ebx

**mov** **ebx,** coefficientOfMultiplyingForTextHeight

;; Convert byte to word

**cbw**

;; coefficientOfMultiplyingForTextHeight \* possibleHeight

;; eax \* ebx

**imul** **ebx**

**imul** **esi**

; print text

PrintInformationInWindow **eax,** TempPlaceForText

**inc** **edi**

**inc** **esi**

**cmp** **edi,** 5

**jne** LoopItself

; free library

invoke FreeLibrary**,** hLib

; create button

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheButton**,** offset TextForOKButton**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_CENTER **or** BS\_TEXT **or** BS\_VCENTER**,**

395**,** 465**,** 150**,** 30**,**

hWnd**,** 7033**,** hInstance**,** NULL

.ELSEIF ourMSG**==**WM\_COMMAND

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSE

; process the message

invoke DefWindowProc**,**hWnd**,**ourMSG**,**wParam**,**lParam

**ret**

.ENDIF

ExitCode**:**

**xor** **eax,** **eax**

**ret**

WndMainProc endp

WndWarnProc proc hWnd**:**HWND**,** ourMSG**:**UINT**,** wParam**:**WPARAM**,** lParam**:**LPARAM

; on window close

.IF ourMSG**==**WM\_CLOSE

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSEIF ourMSG**==**WM\_CREATE

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheButton**,** offset TextForOKButton**,**

WS\_CHILD **or** WS\_VISIBLE **or** BS\_CENTER **or** BS\_TEXT **or** BS\_VCENTER**,**

65**,** 65**,** 150**,** 30**,**

hWnd**,** 7003**,** hInstance**,** NULL

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheText**,** offset StartingText**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_TEXT **or** SS\_CENTER **or** BS\_VCENTER**,**

16**,** 10**,** 250**,** 50**,**

hWnd**,** 7004**,** hInstance**,** NULL

.ELSEIF ourMSG**==**WM\_COMMAND

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSE

; process the message

invoke DefWindowProc**,**hWnd**,**ourMSG**,**wParam**,**lParam

**ret**

.ENDIF

ExitCode**:**

**xor** **eax,eax**

**ret**

WndWarnProc endp

end start

8-9-IP93-Dominskyi-Dynamic-WithoutEntry-Library.asm:

; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

; Libraries And Macroses

;; implementation of FpuFLtoA

includelib **/**masm32**/**lib**/**Fpu.lib

include **/**masm32**/**include**/**Fpu.inc

;; CreateWindowEx, DestroyWindow, PostQuitMessage, DefWindowProc, DispatchMessage && wsprintf

include **/**masm32**/**include**/**user32.inc

includelib **/**masm32**/**lib**/**user32.lib

; ;; implementation of FloatToStr2

include **/**masm32**/**include**/**masm32.inc

includelib **/**masm32**/**lib**/**masm32.lib

.data?

; Start = (2 \* c - d / 23) / (ln(b - a / 4))

; Buffers for final float numbers

BufferFloatA DB 32 DUP**(?)**

BufferFloatB DB 32 DUP**(?)**

BufferFloatC DB 32 DUP**(?)**

BufferFloatD DB 32 DUP**(?)**

BufferFloatFinal DB 32 DUP**(?)**

; Buffers for intermediate results

; First Step

; Value of 2 \* c

BufferTwoMulC DB 32 DUP**(?)**

; Value of d / 23

BufferDdivTwenThree DB 32 DUP**(?)**

; Value of a - 4

BufferAdivFour DB 32 DUP**(?)**

; Second Step

; Value of 2 \* c - d / 23

BufferFirstPart DB 32 DUP**(?)**

; Value of b - a / 4

BufferBsubPartOfLn DB 32 DUP**(?)**

; Third Step

; Value of ln(b - a / 4)

BufferSecondPart DB 32 DUP**(?)**

; Data Segment

.data

ZeroDivisionText DB "Даний вираз має ділення на нуль. Перевірте Свої значення"**,** 13**,** 0

NegativeOrZeroLnText DB "Даний вираз має негативне число або нуль в (ln). Перевірте Свої значення"**,** 13**,** 0

; Name Of Message Box

MsgBoxName DB "8-9-IP93-Dominskyi"**,** 0

firstConstant dq 2.0

secondConstant dq 23.0

thirdConstant dq 4.0

zero dq 0.0

negativeZero dq **-**0.0

;; global variables for interpolating for main window

;; (I will put some int into them and show in main window)

;; mostly used for negative nums

floatFinal DQ 0

; form, which I will be filling with variables

equationVariables DB "For a = (%s), b = (%s), c = (%s) and d = (%s) We have (2 \* (%s) - (%s) / 23) / (ln((%s) - (%s) / 4)) = ((%s) - (%s)) / (ln((%s) - (%s))) = (%s) / (ln((%s))) = (%s) / (%s) = (%s)"**,** 13**,** 0

; Code Segment

.code

; procedure #1 for calculating

DoArithmeticOperations proc aFloat**:** ptr **qword,** bFloat**:** ptr **qword,** cFloat**:** ptr **qword,** dFloat**:** ptr **qword,** TempPlaceForText**:** **dword**

; My equation = (2 \* c - d / 23) / (ln(b - a / 4))

;; values for equation

**mov** **eax,** aFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatA

**mov** **eax,** bFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatB

**mov** **eax,** cFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatC

**mov** **eax,** dFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatD

**finit** ; FPU Initialization

; 2 \* c

; move 2 into st(0)

**fld** firstConstant

; move c into st(0) and 2 into st(1)

**mov** **eax,** cFloat

**fld** **qword** ptr **[eax]**

; multiply 2 by c and move result into st(0)

**fmul**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferTwoMulC**,** SRC1\_FPU **or** SRC2\_DIMM

; d / 23

; move d into st(0) and 2\*c into st(1)

**mov** **eax,** dFloat

**fld** **qword** ptr **[eax]**

; move 23 into st(0), d into st(1) and 2\*c into st(2)

**fld** secondConstant

; divide d by 23 and move result into st(0), 2\*c to st(1)

**fdiv**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferDdivTwenThree**,** SRC1\_FPU **or** SRC2\_DIMM

; 2 \* c - d / 23

; subtract d/23 from 2\*c, move result into st(0)

**fsub**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferFirstPart**,** SRC1\_FPU **or** SRC2\_DIMM

; move ln(2) into st(0) and 2\*c-d/23 into st(1)

**fldln2**

; move b into st(0), ln(2) into st(1) and 2\*c-d/23 into st(2)

**mov** **eax,** bFloat

**fld** **qword** ptr **[eax]**

; move a into st(0), b into st(1), ln(2) into st(2) 2\*c-d/23 into st(3)

**mov** **eax,** aFloat

**fld** **qword** ptr **[eax]**

; move 4 into st(0), a into st(1), b into st(2), ln(2) into st(3) and 2\*c-d/23 into st(4)

**fld** thirdConstant

; divide a by 4 and move it into st(0), b into st(1), ln(2) into st(2) and 2\*c-d/23 into st(3)

**fdiv**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferAdivFour**,** SRC1\_FPU **or** SRC2\_DIMM

; subtract a/4 from b, move result into st(0), ln(2) into st(1) and 2\*c-d/23 into st(2)

**fsub**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferBsubPartOfLn**,** SRC1\_FPU **or** SRC2\_DIMM

; compare, if number is zero or less for ln

; compares the contents of st (0) to the source

**fcom** zero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero

**je** NumberIsLessOrZero

; jump, if less than zero

**jb** NumberIsLessOrZero

; find ln(b - a/4) and move it into st(0), 2\*c-d/23 into st(1)

**fyl2x**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferSecondPart**,** SRC1\_FPU **or** SRC2\_DIMM

; compare, if number is zero for dividing

; compares the contents of st (0) to the source

**fcom** zero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero.zero

**je** NumberIsZero

; compares the contents of st (0) to the source

**fcom** negativeZero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to negative zero.zero

**je** NumberIsZero

; compares the contents of st (0) to zero

**ftst**

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero

**je** NumberIsZero

; divides 2\*c-d/23 by ln(b-a/4) and move it into st(0)

**fdiv**

; (2 \* c - d / 23) / (ln( b - a / 4))

; saves st(0) into variable

**fstp** floatFinal

;; value for final result

invoke FloatToStr2**,** floatFinal**,** **addr** BufferFloatFinal

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** equationVariables**,**

**addr** BufferFloatA**,** **addr** BufferFloatB**,** **addr** BufferFloatC**,** **addr** BufferFloatD**,**

**addr** BufferFloatC**,** **addr** BufferFloatD**,** **addr** BufferFloatB**,** **addr** BufferFloatA**,**

**addr** BufferTwoMulC**,** **addr** BufferDdivTwenThree**,** **addr** BufferFloatB**,**

**addr** BufferAdivFour**,** **addr** BufferFirstPart**,** **addr** BufferBsubPartOfLn**,**

**addr** BufferFirstPart**,** **addr** BufferSecondPart**,** **addr** BufferFloatFinal

**jmp** EndThisMacros

NumberIsZero**:**

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** ZeroDivisionText

**jmp** EndThisMacros

NumberIsLessOrZero**:**

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** NegativeOrZeroLnText

**jmp** EndThisMacros

EndThisMacros**:**

**ret**

DoArithmeticOperations endp

end

8-9-IP93-Dominskyi-Dynamic-WithoutEntry.bat:

:: Our variables (two .asm files, two .obj files, dll and name of the .exe)

**set** NameOfTheFileAsASMParametr**=**"8-9-IP93-Dominskyi-Dynamic-WithoutEntry.asm"

**set** NameOfTheFileAsOBJParametr**=**"8-9-IP93-Dominskyi-Dynamic-WithoutEntry.obj"

**set** NameOfTheFileAsEXEParametr**=**"8-9-IP93-Dominskyi-Dynamic-WithoutEntry.exe"

**set** NameOfTheLibraryAsASMParametr**=**"8-9-IP93-Dominskyi-Dynamic-WithoutEntry-Library.asm"

**set** NameOfTheLibraryAsDLLParametr**=**"8-9-IP93-Dominskyi-Dynamic-WithoutEntry-Library.dll"

**set** NameOfTheLibraryAsOBJParametr**=**"8-9-IP93-Dominskyi-Dynamic-WithoutEntry-Library.obj"

:: There We are combining main file with dll one .exe

:: We can write there, for example, %OurDisk%\masm32\bin\ml, but We have masm commands in environment variables, so need to write only relative path

\masm32\bin\ml /c /coff "**%NameOfTheLibraryAsASMParametr%**"

\masm32\bin\Link.exe /out:"**%NameOfTheLibraryAsDLLParametr%**" /dll /export:DoArithmeticOperations /noentry "**%NameOfTheLibraryAsOBJParametr%**"

\masm32\bin\ml /c /coff "**%NameOfTheFileAsASMParametr%**"

\masm32\bin\Link.exe /subsystem:console "**%NameOfTheFileAsOBJParametr%**"

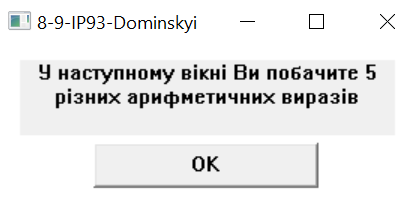
:: if You want window to pause after procedure, then uncomment next row (but You need it, only if You have some problem with code)

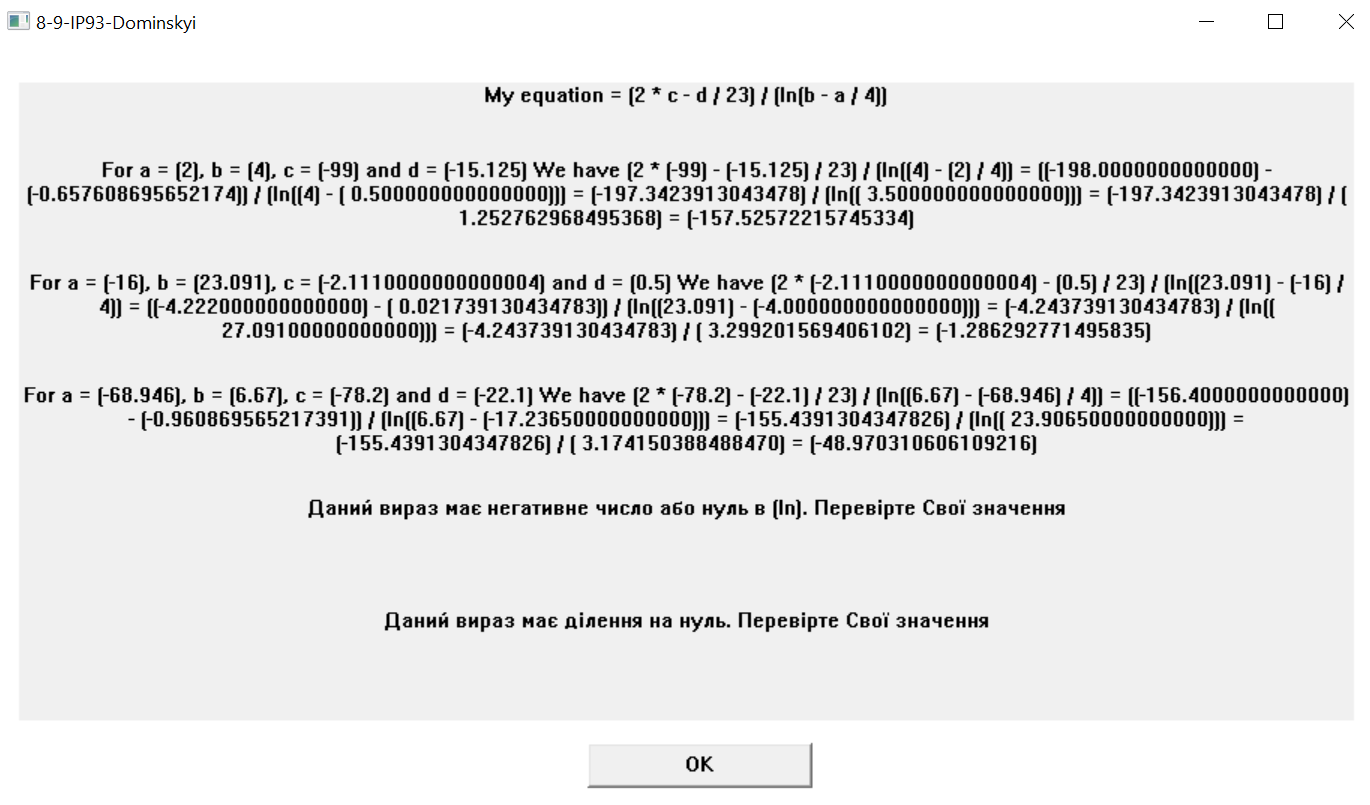
:: pause

:: if You don't want program to run after procedure, then comment next row

**%NameOfTheFileAsEXEParametr%**

Скріншоти:





Статичне підключення з точкою входу:

8-9-IP93-Dominskyi-Static-Entry.asm:

; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

DoArithmeticOperations proto **:**ptr **qword,** **:**ptr **qword,** **:**ptr **qword,** **:**ptr **qword,** **:dword**

WinWarningProto proto **:dword,:dword,:dword**

WinMainProto proto **:dword,:dword,:dword**

; Libraries And Macroses

;; just my library

includelib 8**-**9**-**IP93**-**Dominskyi**-**Static**-**Entry**-**Library.lib

include **/**masm32**/**include**/**windows.inc ;; hInstance, WNDCLASSEX, SW\_SHOWDEFAULT, HWND && NULL

;; CreateWindowEx, DestroyWindow, PostQuitMessage, DefWindowProc, DispatchMessage && wsprintf

include **/**masm32**/**include**/**user32.inc

includelib **/**masm32**/**lib**/**user32.lib

;; GetModuleHandle, ExitProcess, LoadLibrary, GetProcAddress && FreeLibrary

include **/**masm32**/**include**/**kernel32.inc

includelib **/**masm32**/**lib**/**kernel32.lib

; Our Macroses

; We place them here, 'cause it won't degrade the readability of the code

; Macros #1 for printing some text

PrintInformationInWindow macro heightPosition**,** infoToShow

; for example, this commentary is included into macroexpansion

;; but this - not

; just pass position of the text on vertical

; and text, that We want to show

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheText**,** offset infoToShow**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_TEXT **or** SS\_CENTER **or** BS\_VCENTER**,**

16**,** heightPosition**,** 890**,** 75**,**

hWnd**,** 7044**,** hInstance**,** NULL

endm

.data?

hInstance HINSTANCE **?** ; Handle of our program

hWndOfWarnWindow HWND **?** ; Handle of our warn window

hWndOfMainWindow HWND **?** ; Handle of our main window

; Data Segment

.data

StartingText DB "У наступному вікні Ви побачите 5 різних арифметичних виразів"**,** 13**,** 0

TempPlaceForText DB 1024 DUP**(**0**)**

; Name Of Message Box

MsgBoxName DB "8-9-IP93-Dominskyi"**,** 0

NameOfTheWarnWindows DB "Window with warn text"**,** 0 ; the name of our warn window class

NameOfMainWindows DB "Window with main text"**,** 0 ; the name of our success window class

NameOfTheButton DB "Button"**,** 0 ; the name of our button class

NameOfTheText DB "Static"**,** 0 ; the name of our text class

TextForOKButton DB "ОК"**,** 0

; My equation = (2 \* c - d / 23) / (ln( b - a / 4))

; can't be 1 or 0

; first way of declaring array

FloatsA dq 2.0**,** **-**16.0**,** **-**68.946**,** 0.001**,** 4.0 ;; first numbers

FloatsB dq 4.0**,** 23.091**,** 6.67**,** **-**3.33**,** 2.0 ;; second numbers

FloatsC dq **-**99.0**,** **-**2.111**,** **-**78.2**,** 123.4**,** 44.47 ;; third numbers

; and the second one

FloatsD dq **-**15.125 ;; fourth numbers

dq 0.5

dq **-**22.1

dq **-**9.0

dq 12.2222

; for automating

possibleHeight DD 25

coefficientOfMultiplyingForTextHeight DD 3

; first text to show

variantToShow DB "My equation = (2 \* c - d / 23) / (ln(b - a / 4))"**,** 13**,** 0

; Code Segment

.code

start**:** ; Generates program start-up code

invoke WinWarningProto**,** hInstance**,**NULL**,** SW\_SHOWDEFAULT ;invoke function

invoke GetModuleHandle**,** NULL

**mov** hInstance**,** **eax**

invoke WinMainProto**,** hInstance**,**NULL**,** SW\_SHOWDEFAULT ;invoke function

invoke ExitProcess**,** **eax** ; quit program. code returns in EAX register from Main Function.

; function declaration of WinWarn

WinWarningProto proc hInst**:**HINSTANCE**,**hPrevInst**:**HINSTANCE**,**CmdShow**:dword**

; there we need LOCAL variables

LOCAL wc**:**WNDCLASSEX

LOCAL msg**:**MSG

LOCAL hwnd**:**HWND

; assign variables of WNDCLASSEX

; window class is a specification of a window

**mov** wc.cbSize**,** sizeof WNDCLASSEX

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

**mov** wc.lpfnWndProc**,** offset WndWarnProc

**mov** wc.cbClsExtra**,** NULL

**mov** wc.cbWndExtra**,** NULL

**push** hInstance

**pop** wc.hInstance

**mov** wc.hbrBackground**,** COLOR\_WINDOW**+**1

**mov** wc.lpszMenuName**,** NULL

**mov** wc.lpszClassName**,** offset NameOfTheWarnWindows

invoke LoadIcon**,** NULL**,** IDI\_APPLICATION

**mov** wc.hIcon**,** **eax**

**mov** wc.hIconSm**,** **eax**

invoke LoadCursor**,** NULL**,** IDC\_ARROW

**mov** wc.hCursor**,** **eax**

; create class of the window

invoke RegisterClassEx**,** **addr** wc

invoke CreateWindowEx**,** NULL**,**

offset NameOfTheWarnWindows**,**

offset MsgBoxName**,**

WS\_OVERLAPPEDWINDOW **or** DS\_CENTER**,**

470**,** 310**,** 300**,** 150**,**

NULL**,** NULL**,** hInst**,** NULL

**mov** hWndOfWarnWindow**,** **eax**

; write window handle in eax

**mov** hwnd**,eax**

; Show window

invoke ShowWindow**,** hwnd**,**CmdShow

; update screen

invoke UpdateWindow**,** hwnd

; waits for message

.while TRUE

;returns FALSE IF WM\_QUIT message is received and will kill the loop

invoke GetMessage**,** **addr** msg**,**NULL**,**0**,**0

.break .IF **(**!**eax)**

;takes raw keyboard input and generates a new message

invoke TranslateMessage**,** **addr** msg

;sends the message data to the window procedure responsible for the specific window the message is for

invoke DispatchMessage**,** **addr** msg

; end while

.endw

; code returns in EAX register from Main Function.

**mov** **eax,** msg.wParam

; return

**ret**

;The ENDP directive defines the end of the procedure

;and has the same name as in the PROC directive

WinWarningProto endp

; function declaration of WinSuccess

WinMainProto proc hInst**:**HINSTANCE**,**hPrevInst**:**HINSTANCE**,**CmdShow**:dword**

; there we need LOCAL variables

LOCAL wc**:**WNDCLASSEX

LOCAL msg**:**MSG

LOCAL hwnd**:**HWND

; assign variables of WNDCLASSEX

; window class is a specification of a window

**mov** wc.cbSize**,** sizeof WNDCLASSEX

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

**mov** wc.lpfnWndProc**,** offset WndMainProc

**mov** wc.cbClsExtra**,** NULL

**mov** wc.cbWndExtra**,** NULL

**push** hInstance

**pop** wc.hInstance

**mov** wc.hbrBackground**,** COLOR\_WINDOW**+**1

**mov** wc.lpszMenuName**,** NULL

**mov** wc.lpszClassName**,** offset NameOfMainWindows

invoke LoadIcon**,** NULL**,** IDI\_APPLICATION

**mov** wc.hIcon**,** **eax**

**mov** wc.hIconSm**,** **eax**

invoke LoadCursor**,** NULL**,** IDC\_ARROW

**mov** wc.hCursor**,** **eax**

; create class of the window

invoke RegisterClassEx**,** **addr** wc

invoke CreateWindowEx**,** NULL**,**

offset NameOfMainWindows**,**

offset MsgBoxName**,**

WS\_OVERLAPPEDWINDOW **or** DS\_CENTER**,**

170**,** 50**,** 940**,** 550**,**

NULL**,** NULL**,** hInst**,** NULL

**mov** hWndOfMainWindow**,** **eax**

; write window handle in eax

**mov** hwnd**,eax**

; Show window

invoke ShowWindow**,** hwnd**,**CmdShow

; update screen

invoke UpdateWindow**,** hwnd

; waits for message

.while TRUE

;returns FALSE IF WM\_QUIT message is received and will kill the loop

invoke GetMessage**,** **addr** msg**,**NULL**,**0**,**0

.break .IF **(**!**eax)**

;takes raw keyboard input and generates a new message

invoke TranslateMessage**,** **addr** msg

;sends the message data to the window procedure responsible for the specific window the message is for

invoke DispatchMessage**,** **addr** msg

; end while

.endw

; code returns in EAX register from Main Function.

**mov** **eax,** msg.wParam

; return

**ret**

;The ENDP directive defines the end of the procedure

;and has the same name as in the PROC directive

WinMainProto endp

WndMainProc proc hWnd**:**HWND**,** ourMSG**:**UINT**,** wParam**:**WPARAM**,** lParam**:**LPARAM

; on window close

.IF ourMSG**==**WM\_CLOSE

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSEIF ourMSG**==**WM\_CREATE

**mov** **edi,** 0

; invoke macros #1 one time to create text

PrintInformationInWindow possibleHeight**,** offset variantToShow

;; do the loop

LoopItself**:**

invoke DoArithmeticOperations**,** **addr** FloatsA**[**8**\*edi],addr** FloatsB**[**8**\*edi],addr** FloatsC**[**8**\*edi],** **addr** FloatsD**[**8**\*edi],** **addr** TempPlaceForText

; mov possibleHeight into eax

**mov** **eax,** possibleHeight

;; Convert byte to word

**cbw**

; mov possibleHeight into ebx

**mov** **ebx,** coefficientOfMultiplyingForTextHeight

;; Convert byte to word

**cbw**

;; coefficientOfMultiplyingForTextHeight \* possibleHeight

;; eax \* ebx

**imul** **ebx**

**imul** **esi**

; print text

PrintInformationInWindow **eax,** TempPlaceForText

**inc** **edi**

**inc** **esi**

**cmp** **edi,** 5

**jne** LoopItself

; create button

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheButton**,** offset TextForOKButton**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_CENTER **or** BS\_TEXT **or** BS\_VCENTER**,**

395**,** 465**,** 150**,** 30**,**

hWnd**,** 7033**,** hInstance**,** NULL

.ELSEIF ourMSG**==**WM\_COMMAND

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSE

; process the message

invoke DefWindowProc**,**hWnd**,**ourMSG**,**wParam**,**lParam

**ret**

.ENDIF

ExitCode**:**

**xor** **eax,** **eax**

**ret**

WndMainProc endp

WndWarnProc proc hWnd**:**HWND**,** ourMSG**:**UINT**,** wParam**:**WPARAM**,** lParam**:**LPARAM

; on window close

.IF ourMSG**==**WM\_CLOSE

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSEIF ourMSG**==**WM\_CREATE

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheButton**,** offset TextForOKButton**,**

WS\_CHILD **or** WS\_VISIBLE **or** BS\_CENTER **or** BS\_TEXT **or** BS\_VCENTER**,**

65**,** 65**,** 150**,** 30**,**

hWnd**,** 7003**,** hInstance**,** NULL

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheText**,** offset StartingText**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_TEXT **or** SS\_CENTER **or** BS\_VCENTER**,**

16**,** 10**,** 250**,** 50**,**

hWnd**,** 7004**,** hInstance**,** NULL

.ELSEIF ourMSG**==**WM\_COMMAND

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSE

; process the message

invoke DefWindowProc**,**hWnd**,**ourMSG**,**wParam**,**lParam

**ret**

.ENDIF

ExitCode**:**

**xor** **eax,eax**

**ret**

WndWarnProc endp

end start

8-9-IP93-Dominskyi-Static-Entry-Library.asm:

; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

; Libraries And Macroses

;; implementation of FpuFLtoA

includelib **/**masm32**/**lib**/**Fpu.lib

include **/**masm32**/**include**/**Fpu.inc

;; CreateWindowEx, DestroyWindow, PostQuitMessage, DefWindowProc, DispatchMessage && wsprintf

include **/**masm32**/**include**/**user32.inc

includelib **/**masm32**/**lib**/**user32.lib

; ;; implementation of FloatToStr2

include **/**masm32**/**include**/**masm32.inc

includelib **/**masm32**/**lib**/**masm32.lib

.data?

; Start = (2 \* c - d / 23) / (ln(b - a / 4))

; Buffers for final float numbers

BufferFloatA DB 32 DUP**(?)**

BufferFloatB DB 32 DUP**(?)**

BufferFloatC DB 32 DUP**(?)**

BufferFloatD DB 32 DUP**(?)**

BufferFloatFinal DB 32 DUP**(?)**

; Buffers for intermediate results

; First Step

; Value of 2 \* c

BufferTwoMulC DB 32 DUP**(?)**

; Value of d / 23

BufferDdivTwenThree DB 32 DUP**(?)**

; Value of a - 4

BufferAdivFour DB 32 DUP**(?)**

; Second Step

; Value of 2 \* c - d / 23

BufferFirstPart DB 32 DUP**(?)**

; Value of b - a / 4

BufferBsubPartOfLn DB 32 DUP**(?)**

; Third Step

; Value of ln(b - a / 4)

BufferSecondPart DB 32 DUP**(?)**

; Data Segment

.data

ZeroDivisionText DB "Даний вираз має ділення на нуль. Перевірте Свої значення"**,** 13**,** 0

NegativeOrZeroLnText DB "Даний вираз має негативне число або нуль в (ln). Перевірте Свої значення"**,** 13**,** 0

; Name Of Message Box

MsgBoxName DB "8-9-IP93-Dominskyi"**,** 0

firstConstant dq 2.0

secondConstant dq 23.0

thirdConstant dq 4.0

zero dq 0.0

negativeZero dq **-**0.0

;; global variables for interpolating for main window

;; (I will put some int into them and show in main window)

;; mostly used for negative nums

floatFinal DQ 0

; form, which I will be filling with variables

equationVariables DB "For a = (%s), b = (%s), c = (%s) and d = (%s) We have (2 \* (%s) - (%s) / 23) / (ln((%s) - (%s) / 4)) = ((%s) - (%s)) / (ln((%s) - (%s))) = (%s) / (ln((%s))) = (%s) / (%s) = (%s)"**,** 13**,** 0

; Code Segment

.code

DLLmain proc hInstDLL**:** **DWORD,** reason**:** **DWORD,** unused**:** **DWORD**

**mov** **eax,** 1

**ret**

DLLmain endp

; procedure #1 for calculating

DoArithmeticOperations proc aFloat**:** ptr **qword,** bFloat**:** ptr **qword,** cFloat**:** ptr **qword,** dFloat**:** ptr **qword,** TempPlaceForText**:** **dword**

; My equation = (2 \* c - d / 23) / (ln(b - a / 4))

;; values for equation

**mov** **eax,** aFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatA

**mov** **eax,** bFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatB

**mov** **eax,** cFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatC

**mov** **eax,** dFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatD

**finit** ; FPU Initialization

; 2 \* c

; move 2 into st(0)

**fld** firstConstant

; move c into st(0) and 2 into st(1)

**mov** **eax,** cFloat

**fld** **qword** ptr **[eax]**

; multiply 2 by c and move result into st(0)

**fmul**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferTwoMulC**,** SRC1\_FPU **or** SRC2\_DIMM

; d / 23

; move d into st(0) and 2\*c into st(1)

**mov** **eax,** dFloat

**fld** **qword** ptr **[eax]**

; move 23 into st(0), d into st(1) and 2\*c into st(2)

**fld** secondConstant

; divide d by 23 and move result into st(0), 2\*c to st(1)

**fdiv**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferDdivTwenThree**,** SRC1\_FPU **or** SRC2\_DIMM

; 2 \* c - d / 23

; subtract d/23 from 2\*c, move result into st(0)

**fsub**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferFirstPart**,** SRC1\_FPU **or** SRC2\_DIMM

; move ln(2) into st(0) and 2\*c-d/23 into st(1)

**fldln2**

; move b into st(0), ln(2) into st(1) and 2\*c-d/23 into st(2)

**mov** **eax,** bFloat

**fld** **qword** ptr **[eax]**

; move a into st(0), b into st(1), ln(2) into st(2) 2\*c-d/23 into st(3)

**mov** **eax,** aFloat

**fld** **qword** ptr **[eax]**

; move 4 into st(0), a into st(1), b into st(2), ln(2) into st(3) and 2\*c-d/23 into st(4)

**fld** thirdConstant

; divide a by 4 and move it into st(0), b into st(1), ln(2) into st(2) and 2\*c-d/23 into st(3)

**fdiv**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferAdivFour**,** SRC1\_FPU **or** SRC2\_DIMM

; subtract a/4 from b, move result into st(0), ln(2) into st(1) and 2\*c-d/23 into st(2)

**fsub**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferBsubPartOfLn**,** SRC1\_FPU **or** SRC2\_DIMM

; compare, if number is zero or less for ln

; compares the contents of st (0) to the source

**fcom** zero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero

**je** NumberIsLessOrZero

; jump, if less than zero

**jb** NumberIsLessOrZero

; find ln(b - a/4) and move it into st(0), 2\*c-d/23 into st(1)

**fyl2x**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferSecondPart**,** SRC1\_FPU **or** SRC2\_DIMM

; compare, if number is zero for dividing

; compares the contents of st (0) to the source

**fcom** zero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero.zero

**je** NumberIsZero

; compares the contents of st (0) to the source

**fcom** negativeZero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to negative zero.zero

**je** NumberIsZero

; compares the contents of st (0) to zero

**ftst**

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero

**je** NumberIsZero

; divides 2\*c-d/23 by ln(b-a/4) and move it into st(0)

**fdiv**

; (2 \* c - d / 23) / (ln( b - a / 4))

; saves st(0) into variable

**fstp** floatFinal

;; value for final result

invoke FloatToStr2**,** floatFinal**,** **addr** BufferFloatFinal

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** equationVariables**,**

**addr** BufferFloatA**,** **addr** BufferFloatB**,** **addr** BufferFloatC**,** **addr** BufferFloatD**,**

**addr** BufferFloatC**,** **addr** BufferFloatD**,** **addr** BufferFloatB**,** **addr** BufferFloatA**,**

**addr** BufferTwoMulC**,** **addr** BufferDdivTwenThree**,** **addr** BufferFloatB**,**

**addr** BufferAdivFour**,** **addr** BufferFirstPart**,** **addr** BufferBsubPartOfLn**,**

**addr** BufferFirstPart**,** **addr** BufferSecondPart**,** **addr** BufferFloatFinal

**jmp** EndThisMacros

NumberIsZero**:**

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** ZeroDivisionText

**jmp** EndThisMacros

NumberIsLessOrZero**:**

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** NegativeOrZeroLnText

**jmp** EndThisMacros

EndThisMacros**:**

;INVOKE MessageBox, 0, ADDR TempPlaceForText, ADDR MsgBoxName, MB\_OK

**ret**

DoArithmeticOperations endp

end DLLmain

8-9-IP93-Dominskyi-Static-Entry.def:

LIBRARY 8-9-IP93-Dominskyi-Static-Entry-Library

EXPORTS DoArithmeticOperations

8-9-IP93-Dominskyi-Static-Entry.bat:

:: Our variables (two .asm files, two .obj files, .dll, .def and name of the .exe)

**set** NameOfTheFileAsASMParametr**=**"8-9-IP93-Dominskyi-Static-Entry.asm"

**set** NameOfTheFileAsOBJParametr**=**"8-9-IP93-Dominskyi-Static-Entry.obj"

**set** NameOfTheFileAsEXEParametr**=**"8-9-IP93-Dominskyi-Static-Entry.exe"

**set** NameOfTheFileAsDEFParametr**=**"8-9-IP93-Dominskyi-Static-Entry.def"

**set** NameOfTheLibraryAsASMParametr**=**"8-9-IP93-Dominskyi-Static-Entry-Library.asm"

**set** NameOfTheLibraryAsDLLParametr**=**"8-9-IP93-Dominskyi-Static-Entry-Library.dll"

**set** NameOfTheLibraryAsOBJParametr**=**"8-9-IP93-Dominskyi-Static-Entry-Library.obj"

:: There We are combining main file with dll one .exe

:: We can write there, for example, %OurDisk%\masm32\bin\ml, but We have masm commands in environment variables, so need to write only relative path

\masm32\bin\ml /c /coff "**%NameOfTheLibraryAsASMParametr%**"

\masm32\bin\Link.exe /out:"**%NameOfTheLibraryAsDLLParametr%**" /dll /def:**%NameOfTheFileAsDEFParametr%** "**%NameOfTheLibraryAsOBJParametr%**"

\masm32\bin\ml /c /coff "**%NameOfTheFileAsASMParametr%**"

\masm32\bin\Link.exe /subsystem:console "**%NameOfTheFileAsOBJParametr%**"

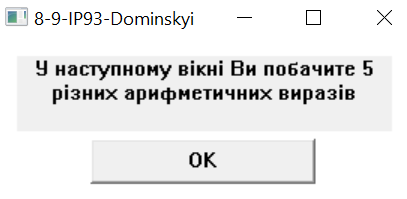
:: if You want window to pause after procedure, then uncomment next row (but You need it, only if You have some problem with code)

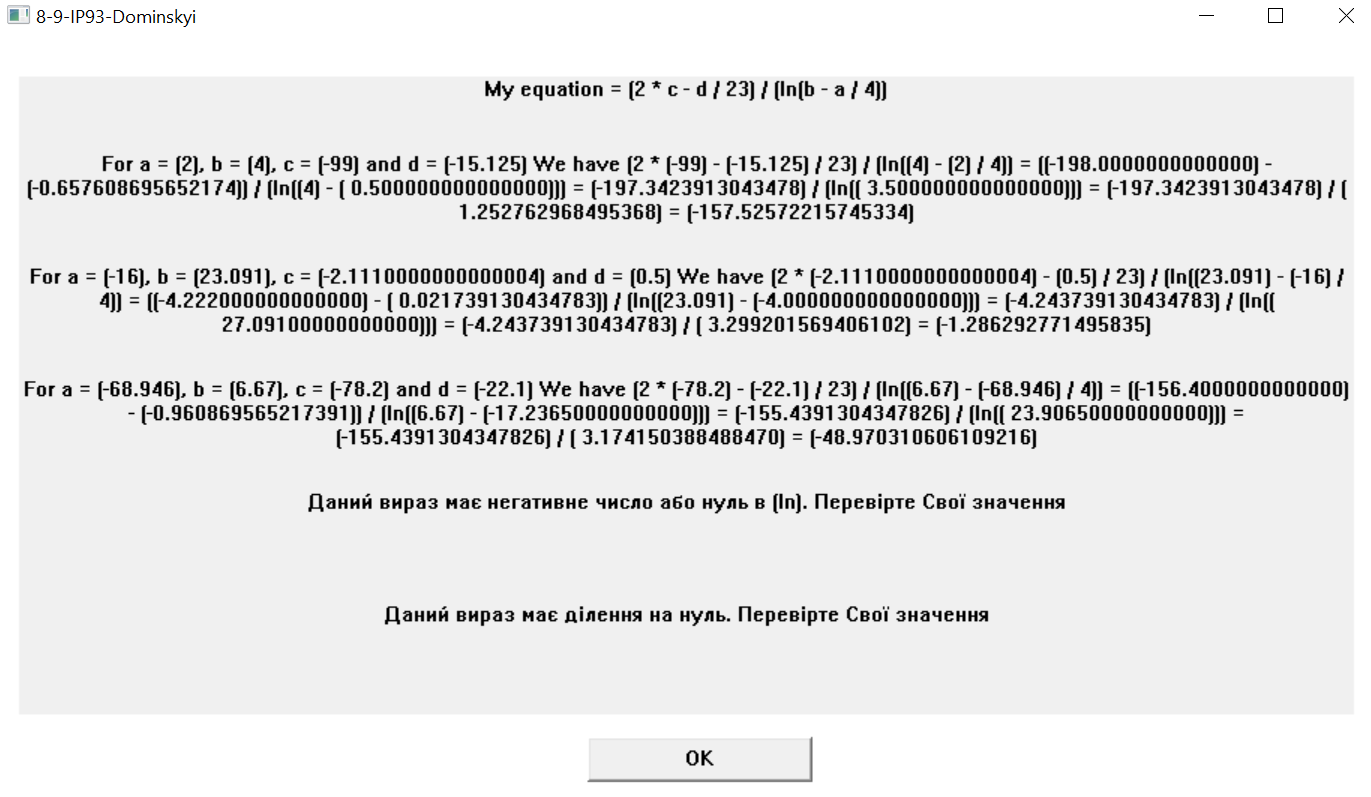
:: pause

:: if You don't want program to run after procedure, then comment next row

**%NameOfTheFileAsEXEParametr%**

Скріншоти:





Статичне підключення без точки входу:

8-9-IP93-Dominskyi-Static-WithoutEntry.asm:

; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

DoArithmeticOperations proto **:**ptr **qword,** **:**ptr **qword,** **:**ptr **qword,** **:**ptr **qword,** **:dword**

WinWarningProto proto **:dword,:dword,:dword**

WinMainProto proto **:dword,:dword,:dword**

; Libraries And Macroses

;; just my library

includelib 8**-**9**-**IP93**-**Dominskyi**-**Static**-**WithoutEntry**-**Library.lib

include **/**masm32**/**include**/**windows.inc ;; hInstance, WNDCLASSEX, SW\_SHOWDEFAULT, HWND && NULL

;; CreateWindowEx, DestroyWindow, PostQuitMessage, DefWindowProc, DispatchMessage && wsprintf

include **/**masm32**/**include**/**user32.inc

includelib **/**masm32**/**lib**/**user32.lib

;; GetModuleHandle, ExitProcess, LoadLibrary, GetProcAddress && FreeLibrary

include **/**masm32**/**include**/**kernel32.inc

includelib **/**masm32**/**lib**/**kernel32.lib

; Our Macroses

; We place them here, 'cause it won't degrade the readability of the code

; Macros #1 for printing some text

PrintInformationInWindow macro heightPosition**,** infoToShow

; for example, this commentary is included into macroexpansion

;; but this - not

; just pass position of the text on vertical

; and text, that We want to show

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheText**,** offset infoToShow**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_TEXT **or** SS\_CENTER **or** BS\_VCENTER**,**

16**,** heightPosition**,** 890**,** 75**,**

hWnd**,** 7044**,** hInstance**,** NULL

endm

.data?

hInstance HINSTANCE **?** ; Handle of our program

hWndOfWarnWindow HWND **?** ; Handle of our warn window

hWndOfMainWindow HWND **?** ; Handle of our main window

; Data Segment

.data

StartingText DB "У наступному вікні Ви побачите 5 різних арифметичних виразів"**,** 13**,** 0

TempPlaceForText DB 1024 DUP**(**0**)**

; Name Of Message Box

MsgBoxName DB "8-9-IP93-Dominskyi"**,** 0

NameOfTheWarnWindows DB "Window with warn text"**,** 0 ; the name of our warn window class

NameOfMainWindows DB "Window with main text"**,** 0 ; the name of our success window class

NameOfTheButton DB "Button"**,** 0 ; the name of our button class

NameOfTheText DB "Static"**,** 0 ; the name of our text class

TextForOKButton DB "OK"**,** 0

; My equation = (2 \* c - d / 23) / (ln( b - a / 4))

; can't be 1 or 0

; first way of declaring array

FloatsA dq 2.0**,** **-**16.0**,** **-**68.946**,** 0.001**,** 4.0 ;; first numbers

FloatsB dq 4.0**,** 23.091**,** 6.67**,** **-**3.33**,** 2.0 ;; second numbers

FloatsC dq **-**99.0**,** **-**2.111**,** **-**78.2**,** 123.4**,** 44.47 ;; third numbers

; and the second one

FloatsD dq **-**15.125 ;; fourth numbers

dq 0.5

dq **-**22.1

dq **-**9.0

dq 12.2222

; for automating

possibleHeight DD 25

coefficientOfMultiplyingForTextHeight DD 3

; first text to show

variantToShow DB "My equation = (2 \* c - d / 23) / (ln(b - a / 4))"**,** 13**,** 0

; Code Segment

.code

start**:** ; Generates program start-up code

invoke WinWarningProto**,** hInstance**,**NULL**,** SW\_SHOWDEFAULT ;invoke function

invoke GetModuleHandle**,** NULL

**mov** hInstance**,** **eax**

invoke WinMainProto**,** hInstance**,**NULL**,** SW\_SHOWDEFAULT ;invoke function

invoke ExitProcess**,** **eax** ; quit program. code returns in EAX register from Main Function.

; function declaration of WinWarn

WinWarningProto proc hInst**:**HINSTANCE**,**hPrevInst**:**HINSTANCE**,**CmdShow**:dword**

; there we need LOCAL variables

LOCAL wc**:**WNDCLASSEX

LOCAL msg**:**MSG

LOCAL hwnd**:**HWND

; assign variables of WNDCLASSEX

; window class is a specification of a window

**mov** wc.cbSize**,** sizeof WNDCLASSEX

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

**mov** wc.lpfnWndProc**,** offset WndWarnProc

**mov** wc.cbClsExtra**,** NULL

**mov** wc.cbWndExtra**,** NULL

**push** hInstance

**pop** wc.hInstance

**mov** wc.hbrBackground**,** COLOR\_WINDOW**+**1

**mov** wc.lpszMenuName**,** NULL

**mov** wc.lpszClassName**,** offset NameOfTheWarnWindows

invoke LoadIcon**,** NULL**,** IDI\_APPLICATION

**mov** wc.hIcon**,** **eax**

**mov** wc.hIconSm**,** **eax**

invoke LoadCursor**,** NULL**,** IDC\_ARROW

**mov** wc.hCursor**,** **eax**

; create class of the window

invoke RegisterClassEx**,** **addr** wc

invoke CreateWindowEx**,** NULL**,**

offset NameOfTheWarnWindows**,**

offset MsgBoxName**,**

WS\_OVERLAPPEDWINDOW **or** DS\_CENTER**,**

470**,** 310**,** 300**,** 150**,**

NULL**,** NULL**,** hInst**,** NULL

**mov** hWndOfWarnWindow**,** **eax**

; write window handle in eax

**mov** hwnd**,eax**

; Show window

invoke ShowWindow**,** hwnd**,**CmdShow

; update screen

invoke UpdateWindow**,** hwnd

; waits for message

.while TRUE

;returns FALSE IF WM\_QUIT message is received and will kill the loop

invoke GetMessage**,** **addr** msg**,**NULL**,**0**,**0

.break .IF **(**!**eax)**

;takes raw keyboard input and generates a new message

invoke TranslateMessage**,** **addr** msg

;sends the message data to the window procedure responsible for the specific window the message is for

invoke DispatchMessage**,** **addr** msg

; end while

.endw

; code returns in EAX register from Main Function.

**mov** **eax,** msg.wParam

; return

**ret**

;The ENDP directive defines the end of the procedure

;and has the same name as in the PROC directive

WinWarningProto endp

; function declaration of WinSuccess

WinMainProto proc hInst**:**HINSTANCE**,**hPrevInst**:**HINSTANCE**,**CmdShow**:dword**

; there we need LOCAL variables

LOCAL wc**:**WNDCLASSEX

LOCAL msg**:**MSG

LOCAL hwnd**:**HWND

; assign variables of WNDCLASSEX

; window class is a specification of a window

**mov** wc.cbSize**,** sizeof WNDCLASSEX

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

**mov** wc.lpfnWndProc**,** offset WndMainProc

**mov** wc.cbClsExtra**,** NULL

**mov** wc.cbWndExtra**,** NULL

**push** hInstance

**pop** wc.hInstance

**mov** wc.hbrBackground**,** COLOR\_WINDOW**+**1

**mov** wc.lpszMenuName**,** NULL

**mov** wc.lpszClassName**,** offset NameOfMainWindows

invoke LoadIcon**,** NULL**,** IDI\_APPLICATION

**mov** wc.hIcon**,** **eax**

**mov** wc.hIconSm**,** **eax**

invoke LoadCursor**,** NULL**,** IDC\_ARROW

**mov** wc.hCursor**,** **eax**

; create class of the window

invoke RegisterClassEx**,** **addr** wc

invoke CreateWindowEx**,** NULL**,**

offset NameOfMainWindows**,**

offset MsgBoxName**,**

WS\_OVERLAPPEDWINDOW **or** DS\_CENTER**,**

170**,** 50**,** 940**,** 550**,**

NULL**,** NULL**,** hInst**,** NULL

**mov** hWndOfMainWindow**,** **eax**

; write window handle in eax

**mov** hwnd**,eax**

; Show window

invoke ShowWindow**,** hwnd**,**CmdShow

; update screen

invoke UpdateWindow**,** hwnd

; waits for message

.while TRUE

;returns FALSE IF WM\_QUIT message is received and will kill the loop

invoke GetMessage**,** **addr** msg**,**NULL**,**0**,**0

.break .IF **(**!**eax)**

;takes raw keyboard input and generates a new message

invoke TranslateMessage**,** **addr** msg

;sends the message data to the window procedure responsible for the specific window the message is for

invoke DispatchMessage**,** **addr** msg

; end while

.endw

; code returns in EAX register from Main Function.

**mov** **eax,** msg.wParam

; return

**ret**

;The ENDP directive defines the end of the procedure

;and has the same name as in the PROC directive

WinMainProto endp

WndMainProc proc hWnd**:**HWND**,** ourMSG**:**UINT**,** wParam**:**WPARAM**,** lParam**:**LPARAM

; on window close

.IF ourMSG**==**WM\_CLOSE

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSEIF ourMSG**==**WM\_CREATE

**mov** **edi,** 0

; invoke macros #1 one time to create text

PrintInformationInWindow possibleHeight**,** offset variantToShow

;; do the loop

LoopItself**:**

invoke DoArithmeticOperations**,** **addr** FloatsA**[**8**\*edi],addr** FloatsB**[**8**\*edi],addr** FloatsC**[**8**\*edi],** **addr** FloatsD**[**8**\*edi],** **addr** TempPlaceForText

; mov possibleHeight into eax

**mov** **eax,** possibleHeight

;; Convert byte to word

**cbw**

; mov possibleHeight into ebx

**mov** **ebx,** coefficientOfMultiplyingForTextHeight

;; Convert byte to word

**cbw**

;; coefficientOfMultiplyingForTextHeight \* possibleHeight

;; eax \* ebx

**imul** **ebx**

**imul** **esi**

; print text

PrintInformationInWindow **eax,** TempPlaceForText

**inc** **edi**

**inc** **esi**

**cmp** **edi,** 5

**jne** LoopItself

; create button

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheButton**,** offset TextForOKButton**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_CENTER **or** BS\_TEXT **or** BS\_VCENTER**,**

395**,** 465**,** 150**,** 30**,**

hWnd**,** 7033**,** hInstance**,** NULL

.ELSEIF ourMSG**==**WM\_COMMAND

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSE

; process the message

invoke DefWindowProc**,**hWnd**,**ourMSG**,**wParam**,**lParam

**ret**

.ENDIF

ExitCode**:**

**xor** **eax,** **eax**

**ret**

WndMainProc endp

WndWarnProc proc hWnd**:**HWND**,** ourMSG**:**UINT**,** wParam**:**WPARAM**,** lParam**:**LPARAM

; on window close

.IF ourMSG**==**WM\_CLOSE

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSEIF ourMSG**==**WM\_CREATE

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheButton**,** offset TextForOKButton**,**

WS\_CHILD **or** WS\_VISIBLE **or** BS\_CENTER **or** BS\_TEXT **or** BS\_VCENTER**,**

65**,** 65**,** 150**,** 30**,**

hWnd**,** 7003**,** hInstance**,** NULL

invoke CreateWindowEx**,**NULL**,**

offset NameOfTheText**,** offset StartingText**,**

WS\_VISIBLE **or** WS\_CHILD **or** BS\_TEXT **or** SS\_CENTER **or** BS\_VCENTER**,**

16**,** 10**,** 250**,** 50**,**

hWnd**,** 7004**,** hInstance**,** NULL

.ELSEIF ourMSG**==**WM\_COMMAND

; exit program

invoke DestroyWindow**,**hWnd

invoke PostQuitMessage**,**NULL

.ELSE

; process the message

invoke DefWindowProc**,**hWnd**,**ourMSG**,**wParam**,**lParam

**ret**

.ENDIF

ExitCode**:**

**xor** **eax,eax**

**ret**

WndWarnProc endp

end start

8-9-IP93-Dominskyi-Static-WithoutEntry-Library.asm:

; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

; Libraries And Macroses

;; implementation of FpuFLtoA

includelib **/**masm32**/**lib**/**Fpu.lib

include **/**masm32**/**include**/**Fpu.inc

;; CreateWindowEx, DestroyWindow, PostQuitMessage, DefWindowProc, DispatchMessage && wsprintf

include **/**masm32**/**include**/**user32.inc

includelib **/**masm32**/**lib**/**user32.lib

; ;; implementation of FloatToStr2

include **/**masm32**/**include**/**masm32.inc

includelib **/**masm32**/**lib**/**masm32.lib

.data?

; Start = (2 \* c - d / 23) / (ln(b - a / 4))

; Buffers for final float numbers

BufferFloatA DB 32 DUP**(?)**

BufferFloatB DB 32 DUP**(?)**

BufferFloatC DB 32 DUP**(?)**

BufferFloatD DB 32 DUP**(?)**

BufferFloatFinal DB 32 DUP**(?)**

; Buffers for intermediate results

; First Step

; Value of 2 \* c

BufferTwoMulC DB 32 DUP**(?)**

; Value of d / 23

BufferDdivTwenThree DB 32 DUP**(?)**

; Value of a - 4

BufferAdivFour DB 32 DUP**(?)**

; Second Step

; Value of 2 \* c - d / 23

BufferFirstPart DB 32 DUP**(?)**

; Value of b - a / 4

BufferBsubPartOfLn DB 32 DUP**(?)**

; Third Step

; Value of ln(b - a / 4)

BufferSecondPart DB 32 DUP**(?)**

; Data Segment

.data

ZeroDivisionText DB "Даний вираз має ділення на нуль. Перевірте Свої значення"**,** 13**,** 0

NegativeOrZeroLnText DB "Даний вираз має негативне число або нуль в (ln). Перевірте Свої значення"**,** 13**,** 0

; Name Of Message Box

MsgBoxName DB "8-9-IP93-Dominskyi"**,** 0

firstConstant dq 2.0

secondConstant dq 23.0

thirdConstant dq 4.0

zero dq 0.0

negativeZero dq **-**0.0

;; global variables for interpolating for main window

;; (I will put some int into them and show in main window)

;; mostly used for negative nums

floatFinal DQ 0

; form, which I will be filling with variables

equationVariables DB "For a = (%s), b = (%s), c = (%s) and d = (%s) We have (2 \* (%s) - (%s) / 23) / (ln((%s) - (%s) / 4)) = ((%s) - (%s)) / (ln((%s) - (%s))) = (%s) / (ln((%s))) = (%s) / (%s) = (%s)"**,** 13**,** 0

; Code Segment

.code

; procedure #1 for calculating

DoArithmeticOperations proc aFloat**:** ptr **qword,** bFloat**:** ptr **qword,** cFloat**:** ptr **qword,** dFloat**:** ptr **qword,** TempPlaceForText**:** **dword**

; My equation = (2 \* c - d / 23) / (ln(b - a / 4))

;; values for equation

**mov** **eax,** aFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatA

**mov** **eax,** bFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatB

**mov** **eax,** cFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatC

**mov** **eax,** dFloat

invoke FloatToStr2**,** **[eax],** **addr** BufferFloatD

**finit** ; FPU Initialization

; 2 \* c

; move 2 into st(0)

**fld** firstConstant

; move c into st(0) and 2 into st(1)

**mov** **eax,** cFloat

**fld** **qword** ptr **[eax]**

; multiply 2 by c and move result into st(0)

**fmul**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferTwoMulC**,** SRC1\_FPU **or** SRC2\_DIMM

; d / 23

; move d into st(0) and 2\*c into st(1)

**mov** **eax,** dFloat

**fld** **qword** ptr **[eax]**

; move 23 into st(0), d into st(1) and 2\*c into st(2)

**fld** secondConstant

; divide d by 23 and move result into st(0), 2\*c to st(1)

**fdiv**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferDdivTwenThree**,** SRC1\_FPU **or** SRC2\_DIMM

; 2 \* c - d / 23

; subtract d/23 from 2\*c, move result into st(0)

**fsub**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferFirstPart**,** SRC1\_FPU **or** SRC2\_DIMM

; move ln(2) into st(0) and 2\*c-d/23 into st(1)

**fldln2**

; move b into st(0), ln(2) into st(1) and 2\*c-d/23 into st(2)

**mov** **eax,** bFloat

**fld** **qword** ptr **[eax]**

; move a into st(0), b into st(1), ln(2) into st(2) 2\*c-d/23 into st(3)

**mov** **eax,** aFloat

**fld** **qword** ptr **[eax]**

; move 4 into st(0), a into st(1), b into st(2), ln(2) into st(3) and 2\*c-d/23 into st(4)

**fld** thirdConstant

; divide a by 4 and move it into st(0), b into st(1), ln(2) into st(2) and 2\*c-d/23 into st(3)

**fdiv**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferAdivFour**,** SRC1\_FPU **or** SRC2\_DIMM

; subtract a/4 from b, move result into st(0), ln(2) into st(1) and 2\*c-d/23 into st(2)

**fsub**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferBsubPartOfLn**,** SRC1\_FPU **or** SRC2\_DIMM

; compare, if number is zero or less for ln

; compares the contents of st (0) to the source

**fcom** zero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero

**je** NumberIsLessOrZero

; jump, if less than zero

**jb** NumberIsLessOrZero

; find ln(b - a/4) and move it into st(0), 2\*c-d/23 into st(1)

**fyl2x**

; convert float to text with 18 digits after "," into buffer

invoke FpuFLtoA**,** 0**,** 18**,** **addr** BufferSecondPart**,** SRC1\_FPU **or** SRC2\_DIMM

; compare, if number is zero for dividing

; compares the contents of st (0) to the source

**fcom** zero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero.zero

**je** NumberIsZero

; compares the contents of st (0) to the source

**fcom** negativeZero

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to negative zero.zero

**je** NumberIsZero

; compares the contents of st (0) to zero

**ftst**

; saves the current value of the SR register to the receiver

**fstsw** **ax**

; loads flags

**sahf**

; jump, if equal to zero

**je** NumberIsZero

; divides 2\*c-d/23 by ln(b-a/4) and move it into st(0)

**fdiv**

; (2 \* c - d / 23) / (ln( b - a / 4))

; saves st(0) into variable

**fstp** floatFinal

;; value for final result

invoke FloatToStr2**,** floatFinal**,** **addr** BufferFloatFinal

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** equationVariables**,**

**addr** BufferFloatA**,** **addr** BufferFloatB**,** **addr** BufferFloatC**,** **addr** BufferFloatD**,**

**addr** BufferFloatC**,** **addr** BufferFloatD**,** **addr** BufferFloatB**,** **addr** BufferFloatA**,**

**addr** BufferTwoMulC**,** **addr** BufferDdivTwenThree**,** **addr** BufferFloatB**,**

**addr** BufferAdivFour**,** **addr** BufferFirstPart**,** **addr** BufferBsubPartOfLn**,**

**addr** BufferFirstPart**,** **addr** BufferSecondPart**,** **addr** BufferFloatFinal

**jmp** EndThisMacros

NumberIsZero**:**

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** ZeroDivisionText

**jmp** EndThisMacros

NumberIsLessOrZero**:**

;; parsing variables into TempPlaceForText

invoke wsprintf**,** TempPlaceForText**,** **addr** NegativeOrZeroLnText

**jmp** EndThisMacros

EndThisMacros**:**

;INVOKE MessageBox, 0, ADDR TempPlaceForText, ADDR MsgBoxName, MB\_OK

**ret**

DoArithmeticOperations endp

end

8-9-IP93-Dominskyi-Static-WithoutEntry.def:

LIBRARY 8-9-IP93-Dominskyi-Static-WithoutEntry-Library

EXPORTS DoArithmeticOperations

8-9-IP93-Dominskyi-Static-WithoutEntry.bat:

:: Our variables (two .asm files, two .obj files, .dll, .def and name of the .exe)

**set** NameOfTheFileAsASMParametr**=**"8-9-IP93-Dominskyi-Static-WithoutEntry.asm"

**set** NameOfTheFileAsOBJParametr**=**"8-9-IP93-Dominskyi-Static-WithoutEntry.obj"

**set** NameOfTheFileAsEXEParametr**=**"8-9-IP93-Dominskyi-Static-WithoutEntry.exe"

**set** NameOfTheFileAsDEFParametr**=**"8-9-IP93-Dominskyi-Static-WithoutEntry.def"

**set** NameOfTheLibraryAsASMParametr**=**"8-9-IP93-Dominskyi-Static-WithoutEntry-Library.asm"

**set** NameOfTheLibraryAsDLLParametr**=**"8-9-IP93-Dominskyi-Static-WithoutEntry-Library.dll"

**set** NameOfTheLibraryAsOBJParametr**=**"8-9-IP93-Dominskyi-Static-WithoutEntry-Library.obj"

:: There We are combining main file with dll one .exe

:: We can write there, for example, %OurDisk%\masm32\bin\ml, but We have masm commands in environment variables, so need to write only relative path

\masm32\bin\ml /c /coff "**%NameOfTheLibraryAsASMParametr%**"

\masm32\bin\Link.exe /out:"**%NameOfTheLibraryAsDLLParametr%**" /dll /noentry /def:**%NameOfTheFileAsDEFParametr%** "**%NameOfTheLibraryAsOBJParametr%**"

\masm32\bin\ml /c /coff "**%NameOfTheFileAsASMParametr%**"

\masm32\bin\Link.exe /subsystem:console "**%NameOfTheFileAsOBJParametr%**"

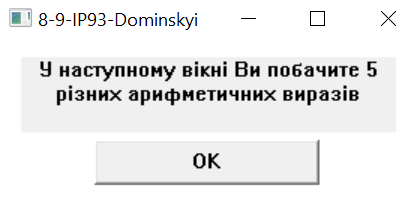
:: if You want window to pause after procedure, then uncomment next row (but You need it, only if You have some problem with code)

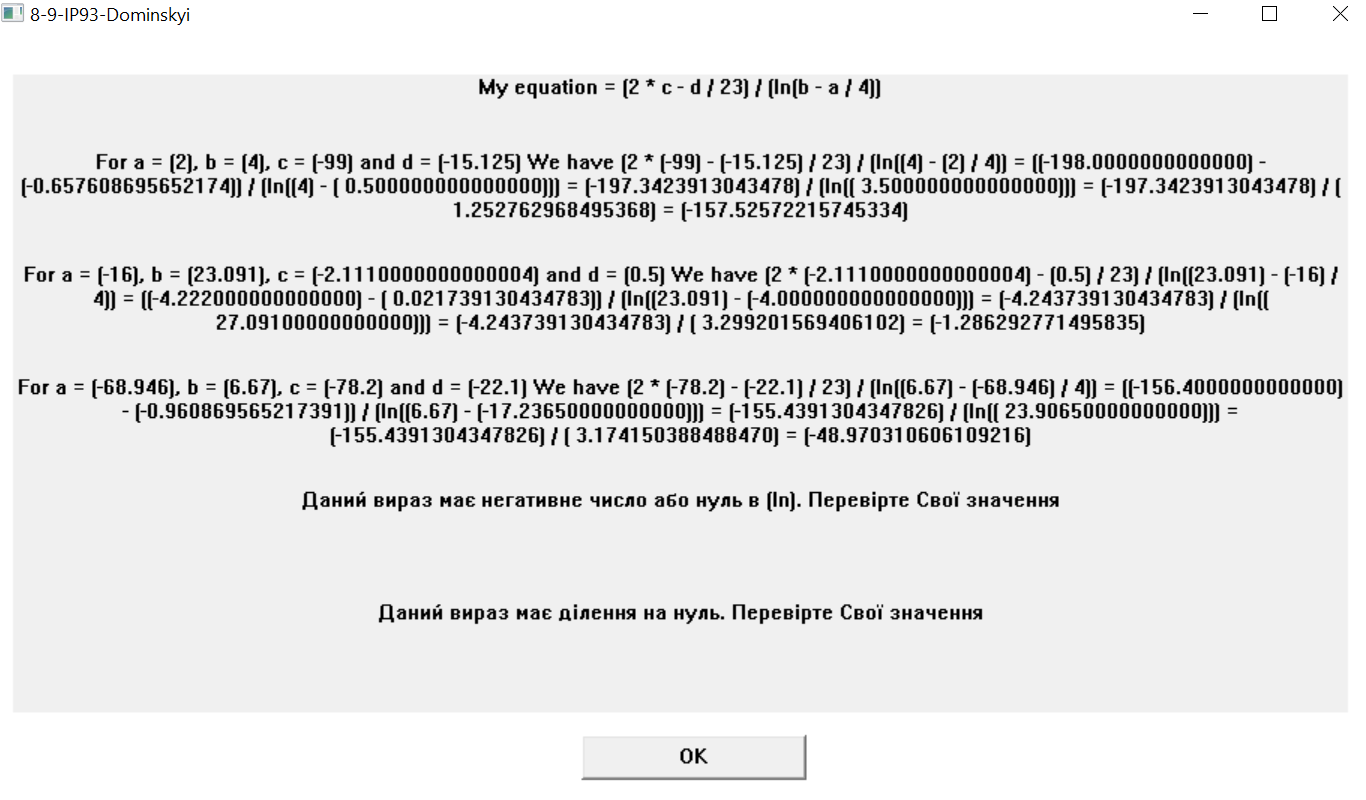
:: pause

:: if You don't want program to run after procedure, then comment next row

**%NameOfTheFileAsEXEParametr%**

Скріншоти:





8.

Номер у списку = 9

Варіант індивідуального завдання:

Спочатку проведемо обчислення з невідомими:

А тепер проведемо для кожних значень a, b, c:

1. a = 2.0, b = 4.0, c = -99.0, d = -15.125
2. a = -16.0, b = 23.091, c = -2.111, d = 0.5
3. a = -68.946, b = 6.67, c = -78.2, d = -22.1
4. a = 0.001, b = -3.33, c = 123.4, d = -9.0

Як видно з даних розрахунків, до ln потрапляє негативне число, що є неможливим для обрахування

1. a = 4.0, b = 2.0, c = 44.47, d = 12.2222

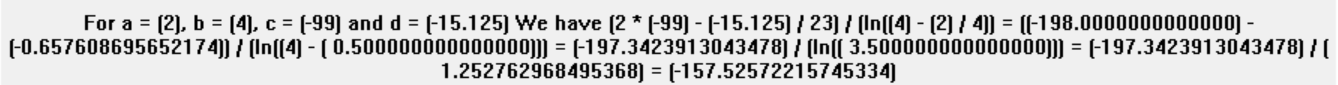
У даному прикладі до знаменнику потрапляє нуль, що є неможливим для обрахування

9.

А тепер давайте звіримо результати з вікна програми та результати, які Ми робили у пункті №5:

Вираз №1:

Результат у вікні:

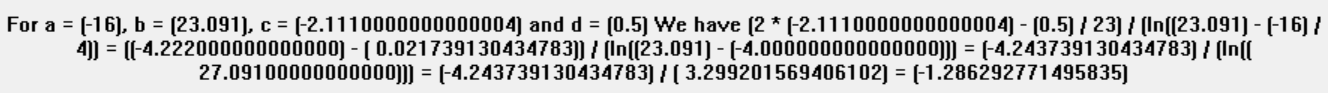


Та той, що Ми вирахували:

1. a = 2.0, b = 4.0, c = -99.0, d = -15.125

Вираз №2:

Результат у вікні:

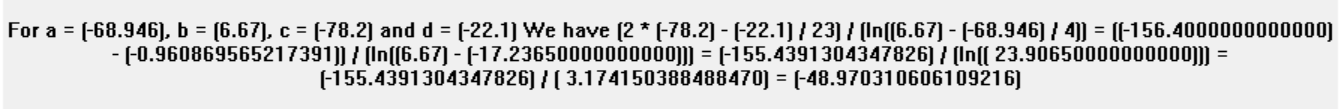


Та той, що Ми вирахували:

1. a = -16.0, b = 23.091, c = -2.111, d = 0.5

Вираз №3:

Результат у вікні:



Та той, що Ми вирахували:

1. a = -68.946, b = 6.67, c = -78.2, d = -22.1

Вираз №4:

Результат у вікні:



Та той, що Ми вирахували:

1. a = 0.001, b = -3.33, c = 123.4, d = -9.0

Як видно з даних розрахунків, до ln потрапляє негативне число, що є неможливим для обрахування

Вираз №5:

Результат у вікні:



Та той, що Ми вирахували:

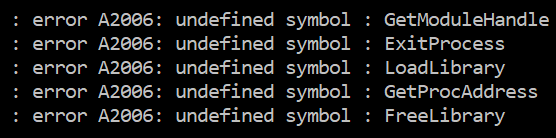
5. a = 4.0, b = 2.0, c = 44.47, d = 12.2222

У даному прикладі до знаменнику потрапляє нуль, що є неможливим для обрахування.

10. Раніше я використував лише одну бібліотеку «masm32rt», але зрозумів, що це є збірником інших бібліотек, тому звідси можна взяти те, що потрібно:

1. kernel32.inc та kernel32.lib використовуються для таких функцій як:

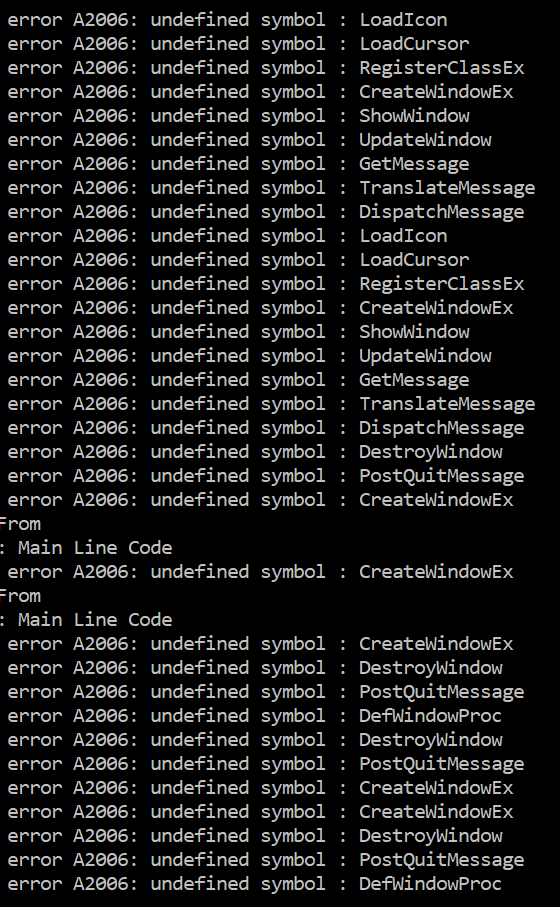
* GetModuleHandle
* ExitProcess
* LoadLibrary
* GetProcAddress
* FreeLibrary



1. user32.inc та user32.lib:

* CreateWindowEx
* DestroyWindow
* PostQuitMessage
* DefWindowProc
* DispatchMessage
* wsprintf

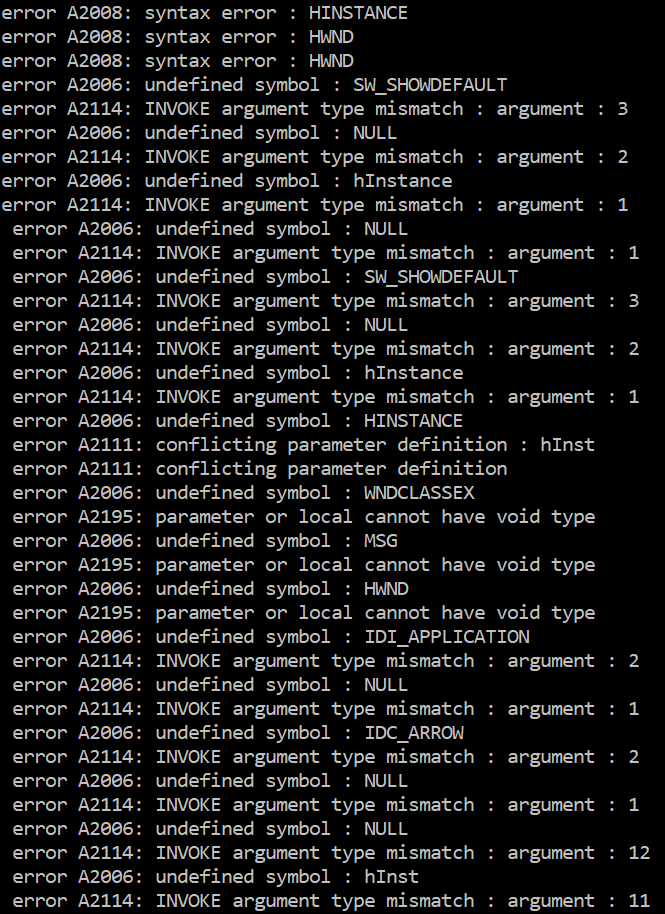
Та багатьох інших подібних функцій:



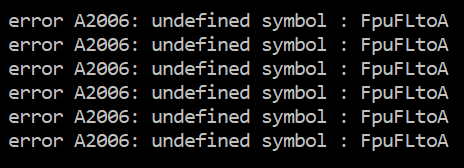
1. Windows.inc:

* hInstance
* WNDCLASSEX
* SW\_SHOWDEFAULT
* HWND
* NULL

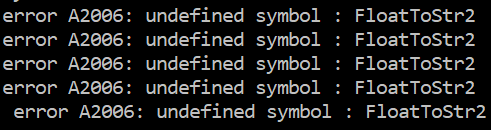
Та інших функцій, які використовуються для створення вікон:



1. Fpu.lib та Fpu.inc потрібні для виклику одного єдиного методу – “FpuFLtoA”, який допомагає з записом чисел до буферів:



1. masm32.inc та masm32.lib – для FLoatToStr2:



**Висновок:**

Я поглибив свої знання роботи з процедурами: написав процедури з використанням точки входу та без вказівки на неї. Зумів завантажити бібліотеки кількома шляхами: явним та неявним. Також познайомився з def файлами. Покращив навички створення bat файлів.