**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 ІМ-21 доцент

Лесько Дмитро Миколайович Павлов В.Г.

Номер у списку групи: 14

**Київ 2024**

**Мета роботи:** Вивчення прийомів розробки і використання процедур, представлених у вигляді динамічних бібліотек.

Формула:



**Виконання програми**

**Неявне з точкою входу**

**Лістинінг коду:**

Головний файл:

.386

.model flat, stdcall

option casemap: none

include \masm32\include\masm32rt.inc

; Needed functions: MessageBox, FloatToStr2, wsprintf, LoadLibrary, GetProcAddress, FreeLibrary, ExitProcess

.data

  zero\_cause\_msg\_box\_title db "Error has appeared", 0

  msg\_box\_title db "Entry dynamic ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)", 0

  zero\_cause\_output\_msg\_box db "Zero appeared into the divider", 10, 13,

      "Our formula with inputed values: ", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "The output of our calculation is undefined, because we can`t divide on zero", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 10, 13,

      "Probably the erro is there: ( d - a + 1). It is the only one place where it can appear", 0

  default\_output\_msg db "Successfuly calculated", 10, 13,

      "Our formula with inputed values: ", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "Result: %s ", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 0

  c\_less\_than\_zero\_msg db "Invalid input values!!!", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "C value can`t be lower than 0!!!", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 10, 13,

      "Input correct params", 0

  debug\_message db "Message is there: %s", 0

  procedure db "defining", 0

  Avalue dq 4.2, 8.1, -1.9, 7.6, 9.8, 55.4

  Bvalue dq -20.4, 12.6, 16.8, 2.2, -1.8, 15.88

  Cvalue dq 2.3, -0.6, 6.5, 9.9, 33.3, 6.1

  Dvalue dq 1.2, -0.5, 0.9, 11.4, 8.8, 3.3

  arraySize equ ($ - Avalue) / 32

  twentyThree dq 23.0

  ten dq 10.0

  four dq 4.0

  minus\_two dq -2.0

  two dq 2.0

  one dq 1.0

  zero dq 0.0

  dll db "8-14-IM-21-Lesko-dll", 0

.data?

  equationBuffer dd 512 dup(?)

  answer dq 8 dup(?)

  fix\_buffer dq 8 dup(?) ; buffer user for debugging the program and tracking the numbers

  divider dt 1 dup(?)

  shared dt 1 dup(?)

  Acurrent db 32 dup(?)

  Bcurrent db 32 dup(?)

  Ccurrent db 32 dup(?)

  Dcurrent db 32 dup(?)

  result db 64 dup(?)

  customLib dd ?

  address dd ?

.code

call\_msg\_window MACRO title:REQ, msg:REQ

    invoke MessageBox, 0, addr msg, addr title, 0

ENDM

main:

  mov esi, 0

  valueLoop:

    cmp esi, arraySize

    jge endLoop

    invoke LoadLibrary, addr dll

    mov customLib, eax

    invoke GetProcAddress, customLib, addr procedure

    mov address, eax

    finit

    invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

    invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

    invoke FloatToStr2, [Bvalue + esi \* 8], addr Bcurrent

    invoke FloatToStr2, [Cvalue + esi \* 8], addr Ccurrent

    invoke FloatToStr2, [Dvalue + esi \* 8], addr Dcurrent

    fld qword ptr [Cvalue + esi \* 8] ; Load the current value of C

    fldz                              ; Load 0

    fcom                               ; Compare C with 0

    fstsw ax                          ; Store the FPU status word in AX

    sahf                              ; Set the flags in EFLAGS register

    ; Test if the C0 (greater than) flag is set

    test ah, 00000001b               ; Check if bit 0 (C0 flag) is set

    jnz continueWithCode             ; If the C0 flag is set, jump to continueWithCode

    ; If C is less than or equal to 0, jump to errorNegativeC

    jmp errorNegativeC

    continueWithCode:

    fld [Avalue + esi \* 8] ; a

    fld [Dvalue + esi \* 8] ; d

    fsub st(0), st(1) ; d - a

    fld one

    fxch st(1)

    fadd st(0), st(1) ; (d - a) + 1

    fstp divider

    fld qword ptr [divider]   ; Load the divider value

    ftst                      ; Compare the divider value with 0

    fnstsw ax                 ; Store the FPU status word in AX

    sahf                      ; Set the flags in EFLAGS register

    test ah, 01000000b       ; Check if bit 7 (C3 flag) is set

    jnz zeroDenominatorError ; If the C3 flag is not set, jump to error handling

    push offset answer

    push offset divider

    push offset shared

    push offset four

    push offset two

    push offset twentyThree

    push offset one

    lea ecx, Dvalue[esi\*8]

    push ecx

    lea ecx, Cvalue[esi\*8]

    push ecx

    lea ecx, Bvalue[esi\*8]

    push ecx

    lea ecx, Avalue[esi\*8]

    push ecx

    call [address]

    ; Format the output message

    invoke FloatToStr2, answer, addr result

    invoke wsprintf, addr equationBuffer, addr default\_output\_msg,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr result,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window msg\_box\_title, equationBuffer

    jmp loopEnd

    errorNegativeC:

    ; Display error message for C, when it is lower than 0

    invoke wsprintf, addr equationBuffer, addr c\_less\_than\_zero\_msg,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

    jmp loopEnd

    zeroDenominatorError:

    ; Display error message for division by zero

    invoke wsprintf, addr equationBuffer, addr zero\_cause\_output\_msg\_box,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

  loopEnd:

    add esi, 1

    jmp valueLoop

  endLoop:

    invoke FreeLibrary, customLib

    invoke ExitProcess, NULL

end main

Файл з процедурою:

.386

.model flat, stdcall

option casemap :none

.code

entryPoint proc hInstDLL: dword, reason:dword

    mov eax, 1

    ret

entryPoint endp

defining proc Avalue: ptr qword, Bvalue: ptr qword, Cvalue: ptr qword, Dvalue: ptr qword,

one: ptr qword, twentyThree: ptr qword, two: ptr qword, four: ptr qword, shared: ptr tbyte, divider: ptr tbyte, answer: ptr qword

    finit

    mov ecx, Avalue  ; define divider

    fld qword ptr [ecx] ; a

    mov ecx, Dvalue

    fld qword ptr [ecx] ; d

    fsub st(0), st(1) ; d - a

    mov ecx, one

    fld qword ptr [ecx] ; 1

    fxch st(1)

    fadd st(0), st(1) ; (d - a) + 1

    mov ecx, divider

    fstp tbyte ptr [ecx]

    mov ecx, one  ; define shared

    fld qword ptr [ecx]

    mov ecx, Cvalue  ; c

    fld qword ptr [ecx]

    fyl2x                                 ; Calculate logarithm base 2 of c: log2(c)

    fldl2t                                ; log2(10)

    fdiv                                  ; Divide log2(c) by log2(10): log2(c) / log2(10) = lg(c)

    mov ecx, four  ; 4

    fld qword ptr [ecx]

    fmul                                  ; Multiply: 4 \* log10(c)

    mov ecx, twentyThree  ; 23

    fld qword ptr [ecx]

    fadd                                  ; Add 23: 4 \* log10(c) + 23

    mov ecx, Bvalue  ; b

    fld qword ptr [ecx]

    mov ecx, two  ; 2

    fld qword ptr [ecx]

    fdiv                                  ; Divide b by 2: b / 2

    fsub                                  ; Subtract b / 2 from 4 \* log10(c) + 23

    mov ecx, shared

    fstp tbyte ptr [ecx] ;; save shared

    mov ecx, shared ; define our answer of this calculation

    fld tbyte ptr [ecx] ; shared (top)

    mov ecx, divider

    fld tbyte ptr [ecx] ; divider (bottom)

    fdivp st(1), st(0) ;shared / divider

    mov ecx, answer

    fstp qword ptr [ecx]

  ret 40

defining endp

end entryPoint

БАТ файл:

\masm32\bin\ml /c /coff "8-14-IM-21-Lesko-dll.asm"

link /out:"8-14-IM-21-Lesko-dll.dll" /export:defining /dll "8-14-IM-21-Lesko-dll.obj"

\masm32\bin\ml /c /coff "8-14-IM-21-Lesko.asm"

link /subsystem:windows "8-14-IM-21-Lesko.obj"

8-14-IM-21-Lesko.exe

**Контрольні розрахунки**:

**Детальні розрахунки контрольних прикладів:**

**Наша формула: ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)**

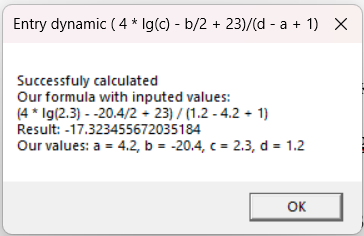
**Перший випадок**:

а = 4.2, б = -20.4, с = 2.3, d = 1.2

Розрахунок:

(4\*lg(2.3)+20.4/2+23)/(1.2-4.2+1)= (4\*0.3617278360175928+10.2+23)/(-2) = 34.64691134407037/-2 = -17.32345567203518

Скріншот з програми:



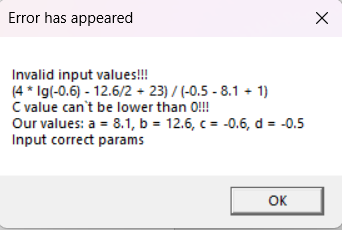
**Другий випадок**:

а = 8.1, б = 12.6, с = -0.6, d = -0.5

Розрахунок:

Цей випадок (приклад) демонструє вдалу валідацію вхідних параметрів нашою програмою. У цьому випадку підставлене число в логарифм менше 0, що порушує його область визначення, тому ми виводимо відповідне повідомлення

Скріншот з програми:



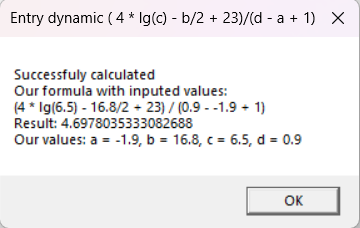
**Третій випадок**:

а = -1.9, б = 16.8, с = 6.5, d = 0.9

Розрахунок:

(4\*lg(6.5)-16.8/2+23)/(0.9+1.9+1)= (4\* 0.8129133566428555-8.4+23)/(3.8) = 17.85165342657142/3.8 = 4.697803533308269

Скріншот з програми:



\*Результат не збігається останніми числами, через округлення калькулятором

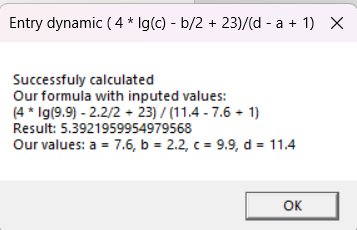
**Четвертий випадок:**

а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(9.9)-2.2/2+23)/(11.4-7.6+1)= (4\* 0.9956351945975499-1.1+23)/(4.8) = 25.8825407783902/4.8 = 5.392195995497958

Скріншот з програми:



**П’ятий випадок(з нулем у знаменнику):**

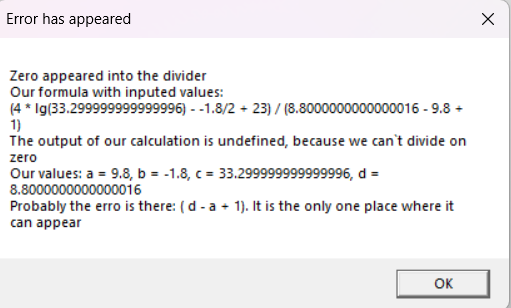
а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(33.3)+1.8/2+23)/(8.8-9.8+1)= 29.98977693402528/ 0 = Infinity?

У цьому випадку ми отримуємо 0 у знаменнику, тому програма не йде далі, а виводить повідомлення про виниклу помилку.

Скріншот з програми:



У такому випадку після виведення повідомлення ми пропускаємо цей цикл і йдемо далі рахувати числа.

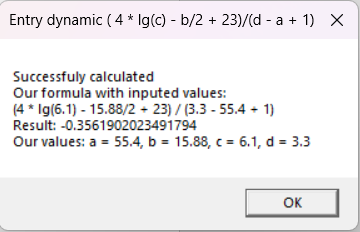
**Шостий випадок:**

а = 55.4, б = 15.88, с = 6.1, d = 3.3

Розрахунок:

(4\*lg(6.1)-15.88/2+23)/(3.3-55.4+1)= (4\* 0.7853298350107669-7.94+23)/( -51.1) = 18.20131934004307/-51.1 = -0.3561902023491794

Скріншот з програми:



**Явне з точкою входу:**

**Лістинінг коду:**

Основний файл:

.386

.model flat, stdcall

option casemap: none

include \masm32\include\masm32rt.inc

; Needed functions: MessageBox, FloatToStr2, wsprintf, ExitProcess

includelib 8-14-IM-21-Lesko-dll.lib

defining proto :ptr qword, :ptr qword, :ptr qword, :ptr qword,

  :ptr qword, :ptr qword, :ptr qword, :ptr qword,

  :ptr tbyte, :ptr tbyte, :ptr qword

.data

  zero\_cause\_msg\_box\_title db "Error has appeared", 0

  msg\_box\_title db "Entry static ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)", 0

  zero\_cause\_output\_msg\_box db "Zero appeared into the divider", 10, 13,

      "Our formula with inputed values: ", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "The output of our calculation is undefined, because we can`t divide on zero", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 10, 13,

      "Probably the erro is there: ( d - a + 1). It is the only one place where it can appear", 0

  default\_output\_msg db "Successfuly calculated", 10, 13,

      "Our formula with inputed values: ", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "Result: %s ", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 0

  c\_less\_than\_zero\_msg db "Invalid input values!!!", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "C value can`t be lower than 0!!!", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 10, 13,

      "Input correct params", 0

  debug\_message db "Message is there: %s", 0

  procedure db "defining", 0

  Avalue dq 4.2, 8.1, -1.9, 7.6, 9.8, 55.4

  Bvalue dq -20.4, 12.6, 16.8, 2.2, -1.8, 15.88

  Cvalue dq 2.3, -0.6, 6.5, 9.9, 33.3, 6.1

  Dvalue dq 1.2, -0.5, 0.9, 11.4, 8.8, 3.3

  arraySize equ ($ - Avalue) / 32

  twentyThree dq 23.0

  ten dq 10.0

  four dq 4.0

  minus\_two dq -2.0

  two dq 2.0

  one dq 1.0

  zero dq 0.0

.data?

  equationBuffer dd 512 dup(?)

  answer dq 8 dup(?)

  fix\_buffer dq 8 dup(?) ; buffer user for debugging the program and tracking the numbers

  divider dt 1 dup(?)

  shared dt 1 dup(?)

  Acurrent db 32 dup(?)

  Bcurrent db 32 dup(?)

  Ccurrent db 32 dup(?)

  Dcurrent db 32 dup(?)

  result db 64 dup(?)

.code

call\_msg\_window MACRO title:REQ, msg:REQ

    invoke MessageBox, 0, addr msg, addr title, 0

ENDM

main:

  mov esi, 0

  valueLoop:

    cmp esi, arraySize

    jge endLoop

    finit

    invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

    invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

    invoke FloatToStr2, [Bvalue + esi \* 8], addr Bcurrent

    invoke FloatToStr2, [Cvalue + esi \* 8], addr Ccurrent

    invoke FloatToStr2, [Dvalue + esi \* 8], addr Dcurrent

    fld qword ptr [Cvalue + esi \* 8] ; Load the current value of C

    fldz                              ; Load 0

    fcom                               ; Compare C with 0

    fstsw ax                          ; Store the FPU status word in AX

    sahf                              ; Set the flags in EFLAGS register

    ; Test if the C0 (greater than) flag is set

    test ah, 00000001b               ; Check if bit 0 (C0 flag) is set

    jnz continueWithCode             ; If the C0 flag is set, jump to continueWithCode

    ; If C is less than or equal to 0, jump to errorNegativeC

    jmp errorNegativeC

    continueWithCode:

    fld [Avalue + esi \* 8] ; a

    fld [Dvalue + esi \* 8] ; d

    fsub st(0), st(1) ; d - a

    fld one

    fxch st(1)

    fadd st(0), st(1) ; (d - a) + 1

    fstp divider

    fld qword ptr [divider]   ; Load the divider value

    ftst                      ; Compare the divider value with 0

    fnstsw ax                 ; Store the FPU status word in AX

    sahf                      ; Set the flags in EFLAGS register

    test ah, 01000000b       ; Check if bit 7 (C3 flag) is set

    jnz zeroDenominatorError ; If the C3 flag is not set, jump to error handling

    invoke defining, addr Avalue[esi\*8], addr Bvalue[esi\*8], addr Cvalue[esi\*8], addr Dvalue[esi\*8],

    addr one, addr twentyThree, addr two, addr four, addr shared, addr divider, addr answer

    ; Format the output message

    invoke FloatToStr2, answer, addr result

    invoke wsprintf, addr equationBuffer, addr default\_output\_msg,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr result,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window msg\_box\_title, equationBuffer

    jmp loopEnd

    errorNegativeC:

    ; Display error message for C, when it is lower than 0

    invoke wsprintf, addr equationBuffer, addr c\_less\_than\_zero\_msg,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

    jmp loopEnd

    zeroDenominatorError:

    ; Display error message for division by zero

    invoke wsprintf, addr equationBuffer, addr zero\_cause\_output\_msg\_box,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

  loopEnd:

    add esi, 1

    jmp valueLoop

  endLoop:

    invoke ExitProcess, NULL

end main

Файл з процедурою:

.386

.model flat, stdcall

option casemap :none

.code

entryPoint proc hInstDLL: dword, reason:dword

    mov eax, 1

    ret

entryPoint endp

defining proc Avalue: ptr qword, Bvalue: ptr qword, Cvalue: ptr qword, Dvalue: ptr qword,

one: ptr qword, twentyThree: ptr qword, two: ptr qword, four: ptr qword, shared: ptr tbyte, divider: ptr tbyte, answer: ptr qword

    finit

    mov ecx, Avalue  ; define divider

    fld qword ptr [ecx] ; a

    mov ecx, Dvalue

    fld qword ptr [ecx] ; d

    fsub st(0), st(1) ; d - a

    mov ecx, one

    fld qword ptr [ecx] ; 1

    fxch st(1)

    fadd st(0), st(1) ; (d - a) + 1

    mov ecx, divider

    fstp tbyte ptr [ecx]

    mov ecx, one  ; define shared

    fld qword ptr [ecx]

    mov ecx, Cvalue  ; c

    fld qword ptr [ecx]

    fyl2x                                 ; Calculate logarithm base 2 of c: log2(c)

    fldl2t                                ; log2(10)

    fdiv                                  ; Divide log2(c) by log2(10): log2(c) / log2(10) = lg(c)

    mov ecx, four  ; 4

    fld qword ptr [ecx]

    fmul                                  ; Multiply: 4 \* log10(c)

    mov ecx, twentyThree  ; 23

    fld qword ptr [ecx]

    fadd                                  ; Add 23: 4 \* log10(c) + 23

    mov ecx, Bvalue  ; b

    fld qword ptr [ecx]

    mov ecx, two  ; 2

    fld qword ptr [ecx]

    fdiv                                  ; Divide b by 2: b / 2

    fsub                                  ; Subtract b / 2 from 4 \* log10(c) + 23

    mov ecx, shared

    fstp tbyte ptr [ecx] ;; save shared

    mov ecx, shared ; define our answer of this calculation

    fld tbyte ptr [ecx] ; shared (top)

    mov ecx, divider

    fld tbyte ptr [ecx] ; divider (bottom)

    fdivp st(1), st(0) ;shared / divider

    mov ecx, answer

    fstp qword ptr [ecx]

  ret

defining endp

end entryPoint

БАТ файл:

\masm32\bin\ml /c /coff "8-14-IM-21-Lesko-dll.asm"

link /out:"8-14-IM-21-Lesko-dll.dll" /dll /def:"8-14-IM-21-Lesko.def" "8-14-IM-21-Lesko-dll.obj"

\masm32\bin\ml /c /coff "8-14-IM-21-Lesko.asm"

link /subsystem:windows "8-14-IM-21-Lesko.obj"

8-14-IM-21-Lesko.exe

ДЕФ файл:

LIBRARY 8-14-IM-21-Lesko-dll

EXPORTS defining

**Контрольні розрахунки**:

**Наша формула: ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)**

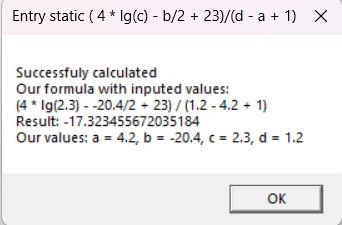
**Перший випадок**:

а = 4.2, б = -20.4, с = 2.3, d = 1.2

Розрахунок:

(4\*lg(2.3)+20.4/2+23)/(1.2-4.2+1)= (4\*0.3617278360175928+10.2+23)/(-2) = 34.64691134407037/-2 = -17.32345567203518

Скріншот з програми:



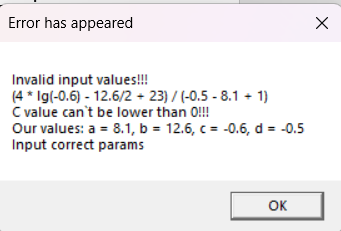
**Другий випадок**:

а = 8.1, б = 12.6, с = -0.6, d = -0.5

Розрахунок:

Цей випадок (приклад) демонструє вдалу валідацію вхідних параметрів нашою програмою. У цьому випадку підставлене число в логарифм менше 0, що порушує його область визначення, тому ми виводимо відповідне повідомлення

Скріншот з програми:



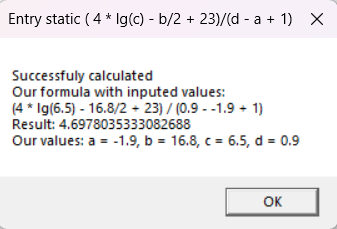
**Третій випадок**:

а = -1.9, б = 16.8, с = 6.5, d = 0.9

Розрахунок:

(4\*lg(6.5)-16.8/2+23)/(0.9+1.9+1)= (4\* 0.8129133566428555-8.4+23)/(3.8) = 17.85165342657142/3.8 = 4.697803533308269

Скріншот з програми:



\*Результат не збігається останніми числами, через округлення калькулятором

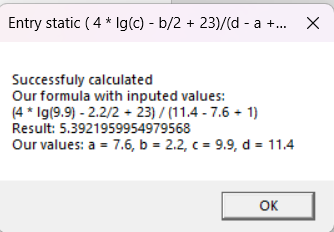
**Четвертий випадок:**

а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(9.9)-2.2/2+23)/(11.4-7.6+1)= (4\* 0.9956351945975499-1.1+23)/(4.8) = 25.8825407783902/4.8 = 5.392195995497958

Скріншот з програми:



**П’ятий випадок(з нулем у знаменнику):**

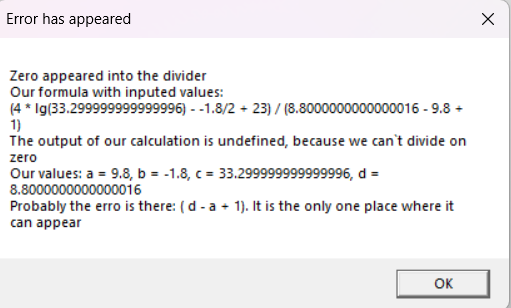
а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(33.3)+1.8/2+23)/(8.8-9.8+1)= 29.98977693402528/ 0 = Infinity?

У цьому випадку ми отримуємо 0 у знаменнику, тому програма не йде далі, а виводить повідомлення про виниклу помилку.

Скріншот з програми:



У такому випадку після виведення повідомлення ми пропускаємо цей цикл і йдемо далі рахувати числа.

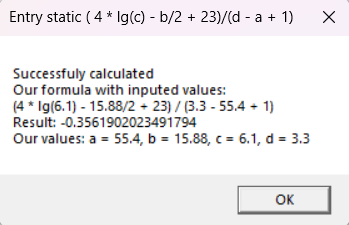
**Шостий випадок:**

а = 55.4, б = 15.88, с = 6.1, d = 3.3

Розрахунок:

(4\*lg(6.1)-15.88/2+23)/(3.3-55.4+1)= (4\* 0.7853298350107669-7.94+23)/( -51.1) = 18.20131934004307/-51.1 = -0.3561902023491794

Скріншот з програми:



**Неявний без точки входу:**

**Лістенінг коду:**

Головний файл:

.386

.model flat, stdcall

option casemap: none

include \masm32\include\masm32rt.inc

; Needed functions: MessageBox, FloatToStr2, wsprintf, LoadLibrary, GetProcAddress, FreeLibrary, ExitProcess

.data

  zero\_cause\_msg\_box\_title db "Error has appeared", 0

  msg\_box\_title db "No entry dynamic ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)", 0

  zero\_cause\_output\_msg\_box db "Zero appeared into the divider", 10, 13,

      "Our formula with inputed values: ", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "The output of our calculation is undefined, because we can`t divide on zero", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 10, 13,

      "Probably the erro is there: ( d - a + 1). It is the only one place where it can appear", 0

  default\_output\_msg db "Successfuly calculated", 10, 13,

      "Our formula with inputed values: ", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "Result: %s ", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 0

  c\_less\_than\_zero\_msg db "Invalid input values!!!", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "C value can`t be lower than 0!!!", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 10, 13,

      "Input correct params", 0

  debug\_message db "Message is there: %s", 0

  procedure db "defining", 0

  Avalue dq 4.2, 8.1, -1.9, 7.6, 9.8, 55.4

  Bvalue dq -20.4, 12.6, 16.8, 2.2, -1.8, 15.88

  Cvalue dq 2.3, -0.6, 6.5, 9.9, 33.3, 6.1

  Dvalue dq 1.2, -0.5, 0.9, 11.4, 8.8, 3.3

  arraySize equ ($ - Avalue) / 32

  twentyThree dq 23.0

  ten dq 10.0

  four dq 4.0

  minus\_two dq -2.0

  two dq 2.0

  one dq 1.0

  zero dq 0.0

  dll db "8-14-IM-21-Lesko-dll", 0

.data?

  equationBuffer dd 512 dup(?)

  answer dq 8 dup(?)

  fix\_buffer dq 8 dup(?) ; buffer user for debugging the program and tracking the numbers

  divider dt 1 dup(?)

  shared dt 1 dup(?)

  Acurrent db 32 dup(?)

  Bcurrent db 32 dup(?)

  Ccurrent db 32 dup(?)

  Dcurrent db 32 dup(?)

  result db 64 dup(?)

  customLib dd ?

  address dd ?

.code

call\_msg\_window MACRO title:REQ, msg:REQ

    invoke MessageBox, 0, addr msg, addr title, 0

ENDM

main:

  mov esi, 0

  valueLoop:

    cmp esi, arraySize

    jge endLoop

    invoke LoadLibrary, addr dll

    mov customLib, eax

    invoke GetProcAddress, customLib, addr procedure

    mov address, eax

    finit

    invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

    invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

    invoke FloatToStr2, [Bvalue + esi \* 8], addr Bcurrent

    invoke FloatToStr2, [Cvalue + esi \* 8], addr Ccurrent

    invoke FloatToStr2, [Dvalue + esi \* 8], addr Dcurrent

    fld qword ptr [Cvalue + esi \* 8] ; Load the current value of C

    fldz                              ; Load 0

    fcom                               ; Compare C with 0

    fstsw ax                          ; Store the FPU status word in AX

    sahf                              ; Set the flags in EFLAGS register

    ; Test if the C0 (greater than) flag is set

    test ah, 00000001b               ; Check if bit 0 (C0 flag) is set

    jnz continueWithCode             ; If the C0 flag is set, jump to continueWithCode

    ; If C is less than or equal to 0, jump to errorNegativeC

    jmp errorNegativeC

    continueWithCode:

    fld [Avalue + esi \* 8] ; a

    fld [Dvalue + esi \* 8] ; d

    fsub st(0), st(1) ; d - a

    fld one

    fxch st(1)

    fadd st(0), st(1) ; (d - a) + 1

    fstp divider

    fld qword ptr [divider]   ; Load the divider value

    ftst                      ; Compare the divider value with 0

    fnstsw ax                 ; Store the FPU status word in AX

    sahf                      ; Set the flags in EFLAGS register

    test ah, 01000000b       ; Check if bit 7 (C3 flag) is set

    jnz zeroDenominatorError ; If the C3 flag is not set, jump to error handling

    push offset answer

    push offset divider

    push offset shared

    push offset four

    push offset two

    push offset twentyThree

    push offset one

    lea ecx, Dvalue[esi\*8]

    push ecx

    lea ecx, Cvalue[esi\*8]

    push ecx

    lea ecx, Bvalue[esi\*8]

    push ecx

    lea ecx, Avalue[esi\*8]

    push ecx

    call [address]

    ; Format the output message

    invoke FloatToStr2, answer, addr result

    invoke wsprintf, addr equationBuffer, addr default\_output\_msg,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr result,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window msg\_box\_title, equationBuffer

    jmp loopEnd

    errorNegativeC:

    ; Display error message for C, when it is lower than 0

    invoke wsprintf, addr equationBuffer, addr c\_less\_than\_zero\_msg,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

    jmp loopEnd

    zeroDenominatorError:

    ; Display error message for division by zero

    invoke wsprintf, addr equationBuffer, addr zero\_cause\_output\_msg\_box,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

  loopEnd:

    add esi, 1

    jmp valueLoop

  endLoop:

    invoke FreeLibrary, customLib

    invoke ExitProcess, NULL

end main

Файл з процедурою:

.386

.model flat, stdcall

option casemap :none

.code

defining proc Avalue: ptr qword, Bvalue: ptr qword, Cvalue: ptr qword, Dvalue: ptr qword,

one: ptr qword, twentyThree: ptr qword, two: ptr qword, four: ptr qword, shared: ptr tbyte, divider: ptr tbyte, answer: ptr qword

    finit

    mov ecx, Avalue  ; define divider

    fld qword ptr [ecx] ; a

    mov ecx, Dvalue

    fld qword ptr [ecx] ; d

    fsub st(0), st(1) ; d - a

    mov ecx, one

    fld qword ptr [ecx] ; 1

    fxch st(1)

    fadd st(0), st(1) ; (d - a) + 1

    mov ecx, divider

    fstp tbyte ptr [ecx]

    mov ecx, one  ; define shared

    fld qword ptr [ecx]

    mov ecx, Cvalue  ; c

    fld qword ptr [ecx]

    fyl2x                                 ; Calculate logarithm base 2 of c: log2(c)

    fldl2t                                ; log2(10)

    fdiv                                  ; Divide log2(c) by log2(10): log2(c) / log2(10) = lg(c)

    mov ecx, four  ; 4

    fld qword ptr [ecx]

    fmul                                  ; Multiply: 4 \* log10(c)

    mov ecx, twentyThree  ; 23

    fld qword ptr [ecx]

    fadd                                  ; Add 23: 4 \* log10(c) + 23

    mov ecx, Bvalue  ; b

    fld qword ptr [ecx]

    mov ecx, two  ; 2

    fld qword ptr [ecx]

    fdiv                                  ; Divide b by 2: b / 2

    fsub                                  ; Subtract b / 2 from 4 \* log10(c) + 23

    mov ecx, shared

    fstp tbyte ptr [ecx] ;; save shared

    mov ecx, shared ; define our answer of this calculation

    fld tbyte ptr [ecx] ; shared (top)

    mov ecx, divider

    fld tbyte ptr [ecx] ; divider (bottom)

    fdivp st(1), st(0) ;shared / divider

    mov ecx, answer

    fstp qword ptr [ecx]

  ret 40

defining endp

end

БАТ файл:

\masm32\bin\ml /c /coff "8-14-IM-21-Lesko-dll.asm"

link /out:"8-14-IM-21-Lesko-dll.dll" /export:defining /dll /noentry "8-14-IM-21-Lesko-dll.obj"

\masm32\bin\ml /c /coff "8-14-IM-21-Lesko.asm"

link /subsystem:windows "8-14-IM-21-Lesko.obj"

8-14-IM-21-Lesko.exe

**Контрольні розрахунки:**

**Наша формула: ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)**

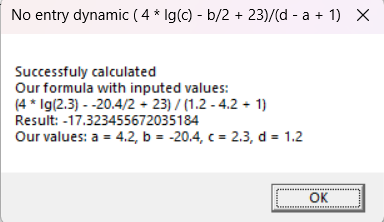
**Перший випадок**:

а = 4.2, б = -20.4, с = 2.3, d = 1.2

Розрахунок:

(4\*lg(2.3)+20.4/2+23)/(1.2-4.2+1)= (4\*0.3617278360175928+10.2+23)/(-2) = 34.64691134407037/-2 = -17.32345567203518

Скріншот з програми:



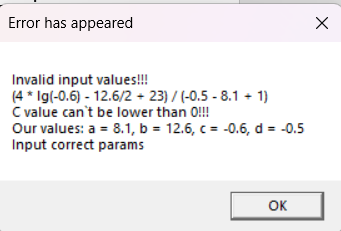
**Другий випадок**:

а = 8.1, б = 12.6, с = -0.6, d = -0.5

Розрахунок:

Цей випадок (приклад) демонструє вдалу валідацію вхідних параметрів нашою програмою. У цьому випадку підставлене число в логарифм менше 0, що порушує його область визначення, тому ми виводимо відповідне повідомлення

Скріншот з програми:



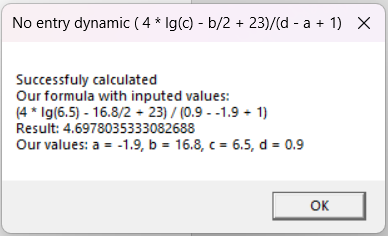
**Третій випадок**:

а = -1.9, б = 16.8, с = 6.5, d = 0.9

Розрахунок:

(4\*lg(6.5)-16.8/2+23)/(0.9+1.9+1)= (4\* 0.8129133566428555-8.4+23)/(3.8) = 17.85165342657142/3.8 = 4.697803533308269

Скріншот з програми:



\*Результат не збігається останніми числами, через округлення калькулятором

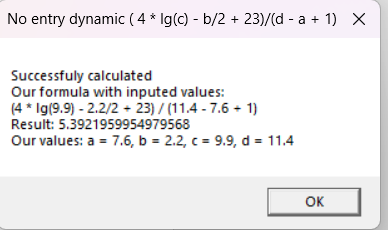
**Четвертий випадок:**

а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(9.9)-2.2/2+23)/(11.4-7.6+1)= (4\* 0.9956351945975499-1.1+23)/(4.8) = 25.8825407783902/4.8 = 5.392195995497958

Скріншот з програми:



**П’ятий випадок(з нулем у знаменнику):**

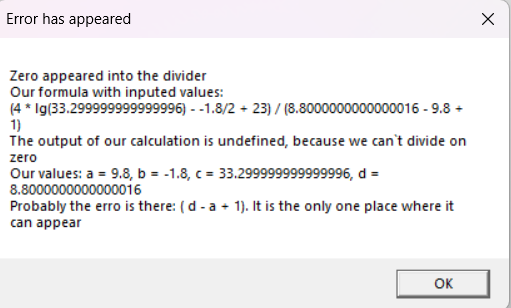
а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(33.3)+1.8/2+23)/(8.8-9.8+1)= 29.98977693402528/ 0 = Infinity?

У цьому випадку ми отримуємо 0 у знаменнику, тому програма не йде далі, а виводить повідомлення про виниклу помилку.

Скріншот з програми:



У такому випадку після виведення повідомлення ми пропускаємо цей цикл і йдемо далі рахувати числа.

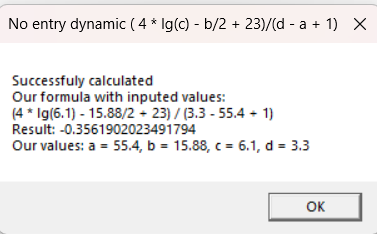
**Шостий випадок:**

а = 55.4, б = 15.88, с = 6.1, d = 3.3

Розрахунок:

(4\*lg(6.1)-15.88/2+23)/(3.3-55.4+1)= (4\* 0.7853298350107669-7.94+23)/( -51.1) = 18.20131934004307/-51.1 = -0.3561902023491794

Скріншот з програми:



**Явне без точки входу**

**Лістенінг коду:**

Основний файл:

.386

.model flat, stdcall

option casemap: none

include \masm32\include\masm32rt.inc

; Needed functions: MessageBox, FloatToStr2, wsprintf, ExitProcess

includelib 8-14-IM-21-Lesko-dll.lib

defining proto :ptr qword, :ptr qword, :ptr qword, :ptr qword,

  :ptr qword, :ptr qword, :ptr qword, :ptr qword,

  :ptr tbyte, :ptr tbyte, :ptr qword

.data

  zero\_cause\_msg\_box\_title db "Error has appeared", 0

  msg\_box\_title db "No entry static ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)", 0

  zero\_cause\_output\_msg\_box db "Zero appeared into the divider", 10, 13,

      "Our formula with inputed values: ", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "The output of our calculation is undefined, because we can`t divide on zero", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 10, 13,

      "Probably the erro is there: ( d - a + 1). It is the only one place where it can appear", 0

  default\_output\_msg db "Successfuly calculated", 10, 13,

      "Our formula with inputed values: ", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "Result: %s ", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 0

  c\_less\_than\_zero\_msg db "Invalid input values!!!", 10, 13,

      "(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

      "C value can`t be lower than 0!!!", 10, 13,

      "Our values: a = %s, b = %s, c = %s, d = %s", 10, 13,

      "Input correct params", 0

  debug\_message db "Message is there: %s", 0

  procedure db "defining", 0

  Avalue dq 4.2, 8.1, -1.9, 7.6, 9.8, 55.4

  Bvalue dq -20.4, 12.6, 16.8, 2.2, -1.8, 15.88

  Cvalue dq 2.3, -0.6, 6.5, 9.9, 33.3, 6.1

  Dvalue dq 1.2, -0.5, 0.9, 11.4, 8.8, 3.3

  arraySize equ ($ - Avalue) / 32

  twentyThree dq 23.0

  ten dq 10.0

  four dq 4.0

  minus\_two dq -2.0

  two dq 2.0

  one dq 1.0

  zero dq 0.0

.data?

  equationBuffer dd 512 dup(?)

  answer dq 8 dup(?)

  fix\_buffer dq 8 dup(?) ; buffer user for debugging the program and tracking the numbers

  divider dt 1 dup(?)

  shared dt 1 dup(?)

  Acurrent db 32 dup(?)

  Bcurrent db 32 dup(?)

  Ccurrent db 32 dup(?)

  Dcurrent db 32 dup(?)

  result db 64 dup(?)

.code

call\_msg\_window MACRO title:REQ, msg:REQ

    invoke MessageBox, 0, addr msg, addr title, 0

ENDM

main:

  mov esi, 0

  valueLoop:

    cmp esi, arraySize

    jge endLoop

    finit

    invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

    invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

    invoke FloatToStr2, [Bvalue + esi \* 8], addr Bcurrent

    invoke FloatToStr2, [Cvalue + esi \* 8], addr Ccurrent

    invoke FloatToStr2, [Dvalue + esi \* 8], addr Dcurrent

    fld qword ptr [Cvalue + esi \* 8] ; Load the current value of C

    fldz                              ; Load 0

    fcom                               ; Compare C with 0

    fstsw ax                          ; Store the FPU status word in AX

    sahf                              ; Set the flags in EFLAGS register

    ; Test if the C0 (greater than) flag is set

    test ah, 00000001b               ; Check if bit 0 (C0 flag) is set

    jnz continueWithCode             ; If the C0 flag is set, jump to continueWithCode

    ; If C is less than or equal to 0, jump to errorNegativeC

    jmp errorNegativeC

    continueWithCode:

    fld [Avalue + esi \* 8] ; a

    fld [Dvalue + esi \* 8] ; d

    fsub st(0), st(1) ; d - a

    fld one

    fxch st(1)

    fadd st(0), st(1) ; (d - a) + 1

    fstp divider

    fld qword ptr [divider]   ; Load the divider value

    ftst                      ; Compare the divider value with 0

    fnstsw ax                 ; Store the FPU status word in AX

    sahf                      ; Set the flags in EFLAGS register

    test ah, 01000000b       ; Check if bit 7 (C3 flag) is set

    jnz zeroDenominatorError ; If the C3 flag is not set, jump to error handling

    invoke defining, addr Avalue[esi\*8], addr Bvalue[esi\*8], addr Cvalue[esi\*8], addr Dvalue[esi\*8],

    addr one, addr twentyThree, addr two, addr four, addr shared, addr divider, addr answer

    ; Format the output message

    invoke FloatToStr2, answer, addr result

    invoke wsprintf, addr equationBuffer, addr default\_output\_msg,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr result,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window msg\_box\_title, equationBuffer

    jmp loopEnd

    errorNegativeC:

    ; Display error message for C, when it is lower than 0

    invoke wsprintf, addr equationBuffer, addr c\_less\_than\_zero\_msg,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

    jmp loopEnd

    zeroDenominatorError:

    ; Display error message for division by zero

    invoke wsprintf, addr equationBuffer, addr zero\_cause\_output\_msg\_box,

        addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

        addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

    call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

  loopEnd:

    add esi, 1

    jmp valueLoop

  endLoop:

    invoke ExitProcess, NULL

end main

Файл процедури:

.386

.model flat, stdcall

option casemap :none

.code

defining proc Avalue: ptr qword, Bvalue: ptr qword, Cvalue: ptr qword, Dvalue: ptr qword,

one: ptr qword, twentyThree: ptr qword, two: ptr qword, four: ptr qword, shared: ptr tbyte, divider: ptr tbyte, answer: ptr qword

finit

mov ecx, Avalue ; define divider

fld qword ptr [ecx] ; a

mov ecx, Dvalue

fld qword ptr [ecx] ; d

fsub st(0), st(1) ; d - a

mov ecx, one

fld qword ptr [ecx] ; 1

fxch st(1)

fadd st(0), st(1) ; (d - a) + 1

mov ecx, divider

fstp tbyte ptr [ecx]

mov ecx, one ; define shared

fld qword ptr [ecx]

mov ecx, Cvalue ; c

fld qword ptr [ecx]

fyl2x ; Calculate logarithm base 2 of c: log2(c)

fldl2t ; log2(10)

fdiv ; Divide log2(c) by log2(10): log2(c) / log2(10) = lg(c)

mov ecx, four ; 4

fld qword ptr [ecx]

fmul ; Multiply: 4 \* log10(c)

mov ecx, twentyThree ; 23

fld qword ptr [ecx]

fadd ; Add 23: 4 \* log10(c) + 23

mov ecx, Bvalue ; b

fld qword ptr [ecx]

mov ecx, two ; 2

fld qword ptr [ecx]

fdiv ; Divide b by 2: b / 2

fsub ; Subtract b / 2 from 4 \* log10(c) + 23

mov ecx, shared

fstp tbyte ptr [ecx] ;; save shared

mov ecx, shared ; define our answer of this calculation

fld tbyte ptr [ecx] ; shared (top)

mov ecx, divider

fld tbyte ptr [ecx] ; divider (bottom)

fdivp st(1), st(0) ;shared / divider

mov ecx, answer

fstp qword ptr [ecx]

ret

defining endp

end

БАТ файл:

\masm32\bin\ml /c /coff "8-14-IM-21-Lesko-dll.asm"

link /out:"8-14-IM-21-Lesko-dll.dll" /def:"8-14-IM-21-Lesko.def" /dll /noentry "8-14-IM-21-Lesko-dll.obj"

\masm32\bin\ml /c /coff "8-14-IM-21-Lesko.asm"

link /subsystem:windows "8-14-IM-21-Lesko.obj"

8-14-IM-21-Lesko.exe

ДЕФ файл:

LIBRARY 8-14-IM-21-Lesko-dll

EXPORTS defining

**Контрольні розрахунки**:

**Наша формула: ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)**

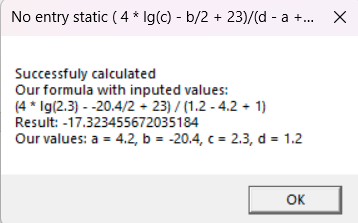
**Перший випадок**:

а = 4.2, б = -20.4, с = 2.3, d = 1.2

Розрахунок:

(4\*lg(2.3)+20.4/2+23)/(1.2-4.2+1)= (4\*0.3617278360175928+10.2+23)/(-2) = 34.64691134407037/-2 = -17.32345567203518

Скріншот з програми:



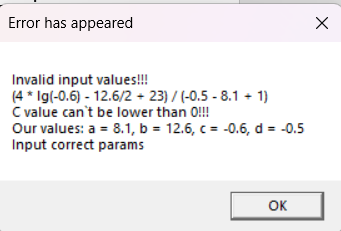
**Другий випадок**:

а = 8.1, б = 12.6, с = -0.6, d = -0.5

Розрахунок:

Цей випадок (приклад) демонструє вдалу валідацію вхідних параметрів нашою програмою. У цьому випадку підставлене число в логарифм менше 0, що порушує його область визначення, тому ми виводимо відповідне повідомлення

Скріншот з програми:



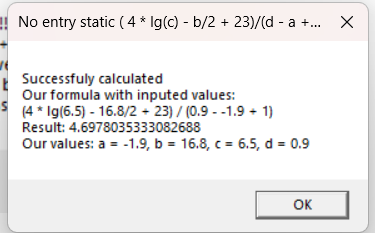
**Третій випадок**:

а = -1.9, б = 16.8, с = 6.5, d = 0.9

Розрахунок:

(4\*lg(6.5)-16.8/2+23)/(0.9+1.9+1)= (4\* 0.8129133566428555-8.4+23)/(3.8) = 17.85165342657142/3.8 = 4.697803533308269

Скріншот з програми:



\*Результат не збігається останніми числами, через округлення калькулятором

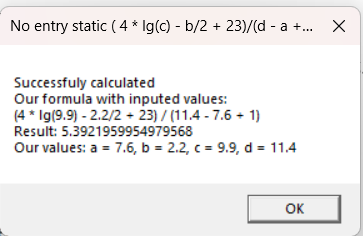
**Четвертий випадок:**

а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(9.9)-2.2/2+23)/(11.4-7.6+1)= (4\* 0.9956351945975499-1.1+23)/(4.8) = 25.8825407783902/4.8 = 5.392195995497958

Скріншот з програми:



**П’ятий випадок(з нулем у знаменнику):**

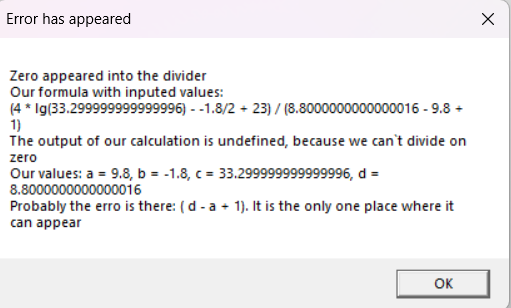
а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(33.3)+1.8/2+23)/(8.8-9.8+1)= 29.98977693402528/ 0 = Infinity?

У цьому випадку ми отримуємо 0 у знаменнику, тому програма не йде далі, а виводить повідомлення про виниклу помилку.

Скріншот з програми:



У такому випадку після виведення повідомлення ми пропускаємо цей цикл і йдемо далі рахувати числа.

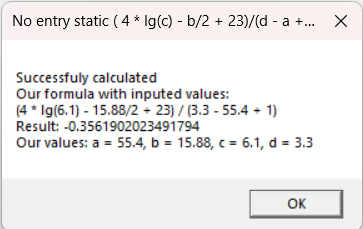
**Шостий випадок:**

а = 55.4, б = 15.88, с = 6.1, d = 3.3

Розрахунок:

(4\*lg(6.1)-15.88/2+23)/(3.3-55.4+1)= (4\* 0.7853298350107669-7.94+23)/( -51.1) = 18.20131934004307/-51.1 = -0.3561902023491794

Скріншот з програми:



**Висновок:**

Під час виконання цієї лабораторної роботи я детально дослідив процеси розробки та застосування динамічних бібліотек (DLL). Я опанував різні методи створення DLL-бібліотек та їхнього використання, зокрема, з і без вказівки точки входу. Для перевірки правильності виконаних обчислень я здійснив контрольні розрахунки і порівняв їх з результатами, отриманими від програми. Також було проведено налагодження програми шляхом порівняння результатів з контрольними прикладами.

Під час виконання роботи зіштовхнувся з багатьма труднощами, адже хоч досліджуванні підходи дають неймовірно великі можливості у підключенні бібліотек, так і у виділенні модулів програми, все-таки програма починає складатися з багатьох файлів, тим самим ускладнюючи процес її білдингу, що змушує шукати рішення.

Я дійшов таких висновків: вивчення та використання динамічних бібліотек є важливим аспектом системного програмування, вони забезпечують зручність та гнучкість у розробці програмного забезпечення, дозволяючи повторно використовувати функціональні блоки та процедури.

Результат- збільшення можливостей підключення бібліотек, що в різних ситуаціях може бути в нагоді. Загалом зручно мати декілька підходів, коли якийсь з них не може бути реалізованим, а вони, на щастя, є взамозамінними.

Завдяки цій лабораторній роботі я отримав практичні навички розробки та інтеграції динамічних бібліотек. Мною були успішно реалізовані процедури обчислення заданих арифметичних виразів та виведення результатів за допомою операцій з плаваючою комою. Також було проведено перевірку правильності обчислень за допомогою контрольних розрахунків, які вдало збіглися, дані інколи були більш повноцінні ніж такі, отримані за допомогою просунутих онлайн калькуляторів, що ще раз збільшує авторитетність цієї лабораторної роботи.