**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 №7**

з дисципліни  
 «Системне програмування»

на тему

“Модульне програмування. Використання процедур”

Виконав: Перевірив:

Студент 2-го курсу групи ІМ-13 Павлов Валерій Георгійович

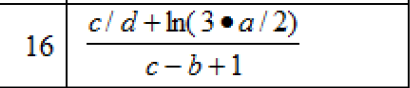
Нестеров Дмитро Васильович  
номер у списку групи: 16

Номер залікової книжки:IM-1320

**Київ 2023**

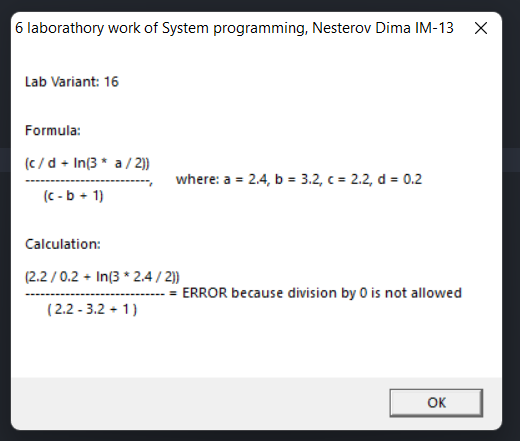
**Мета роботи**

Вивчення прийомів модульного програмування, методів звернення до процедур і передачі в них параметрів.

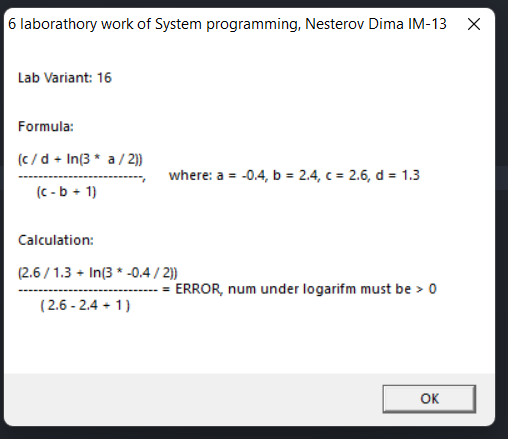


**Скріншоти виконання програми**

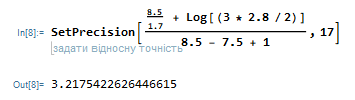
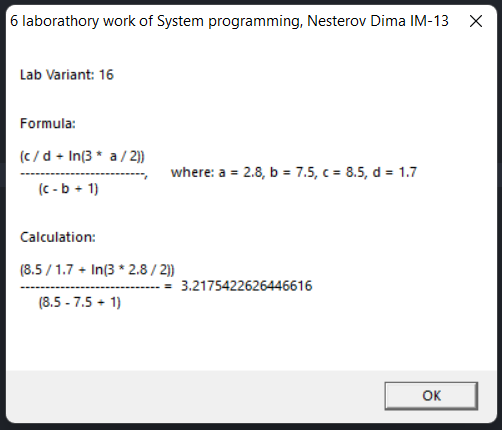
Я зрозумів, що у минулих двох лабораторних були зняті бали за використання заборонених значень у прикладах, тому виправив це у цій лабораторній роботі. Я зробив 6 прикладів, два з яких перевіряють виключні ситуації, а інші чотири варіанти перевіряють правильність обрахунків для всіх комбінацій знаків чисельника і знаменника головного дробу. Числа з якими оперувала програма не змінилися, результат виконання також, що доводить, що програма рахує правильно і параметри у процедури передавалися вірно.



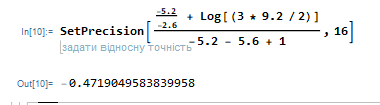
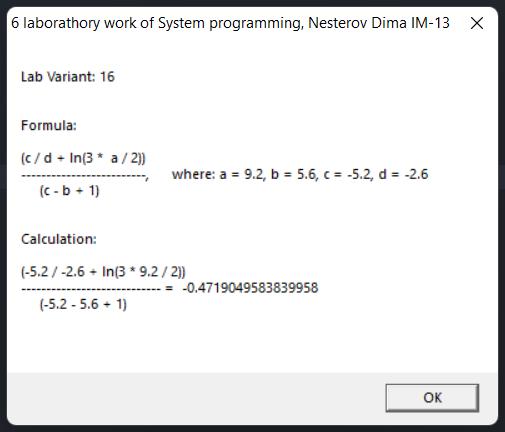
В даному випадку 2.2 – 3.2 + 1 = 0, отже ми знову маємо виключну ситуацію при знаменнику, що дорівнює нулю.



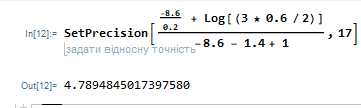
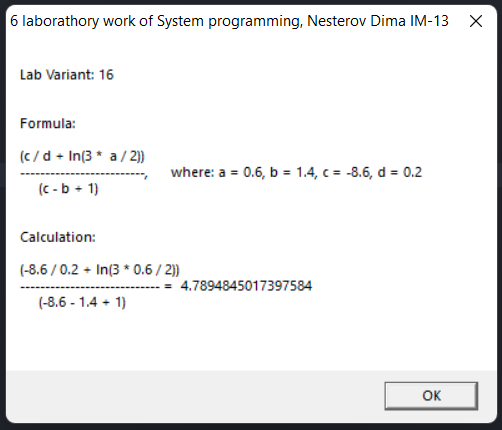
Знову виключна ситуація. У цьому випадку 3 \* (-0.4) / 2 = - 0.6, що менше нуля, що не входить в область визначення логарифму.



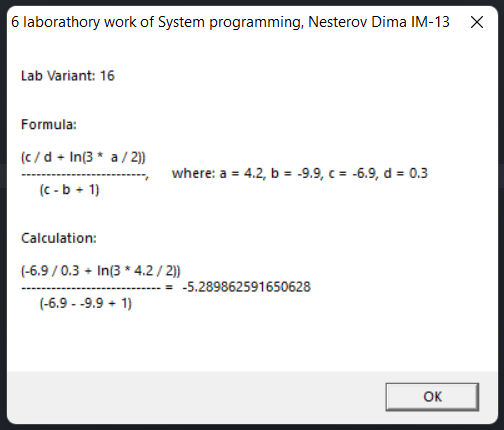
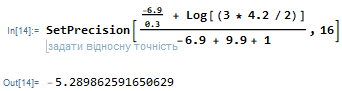
3.2175422626446615 3.2175422626446616, відрізняється лише остання цифра.



-0.4719049583839958 -0.4719049583839958, співпало повністю.



4.7894845017397580 4.7894845017397584, відрізняється лише остання цифра.

- 5.289862591650629 - 5.289862591650628, відрізняється лише остання цифра.

**Лістинг програми**

.386

.MODEL flat, stdcall

OPTION casemap :none

INCLUDE \masm32\include\masm32rt.inc

PUBLIC NESTER\_ARRAY\_C, NESTER\_ARRAY\_B, NESTER\_CONST1

EXTERN NESTER\_THIRD\_PROC:PROTO

.DATA

NESTER\_ARRAY\_A dq 2.4, -0.4, 2.8, 9.2, 0.6, 4.2

NESTER\_ARRAY\_B dq 3.2, 2.4, 7.5, 5.6, 1.4, -9.9 ;;my arrays with a b с and d =)

NESTER\_ARRAY\_C dq 2.2, 2.6, 8.5, -5.2, -8.6, -6.9

NESTER\_ARRAY\_D dq 0.2, 1.3, 1.7, -2.6, 0.2, 0.3

NESTER\_RESULT dq 0.0

NESTER\_CONST3 dq 3.0

NESTER\_CONST2 dq 2.0

NESTER\_CONST1 dq 1.0

NESTER\_TEXT\_ARRAY\_A db 64 dup(?)

NESTER\_TEXT\_ARRAY\_B db 64 dup(?)

NESTER\_TEXT\_ARRAY\_C db 64 dup(?)

NESTER\_TEXT\_ARRAY\_D db 64 dup(?)

NESTER\_TEXT\_RESULT db 32 dup(?)

NESTER\_MESSAGE db 512 dup(?)

NESTER\_WINDOW\_TITLE db "6 laborathory work of System programming, Nesterov Dima IM-13", 0

NESTER\_MESSAGE\_FORMAT db "Lab Variant: 16", 10, 10, 10,

"Formula:", 10, 10, "(c / d + ln(3 \* a / 2))", 10, "-------------------------,", " where: a = %s, b = %s, c = %s, d = %s", 10, " (c - b + 1)", 10, 10, 10,

"Calculation:", 10, 10,

"(%s / %s + ln(3 \* %s / 2))", 10, "---------------------------- = %s", 10, " (%s - %s + 1)", 10, 10, 10,0

NESTER\_ZERO\_DENOMITATOR\_FORMAT db "Lab Variant: 16", 10, 10, 10,

"Formula:", 10, 10, "(c / d + ln(3 \* a / 2))", 10, "-------------------------,", " where: a = %s, b = %s, c = %s, d = %s", 10, " (c - b + 1)", 10, 10, 10,

"Calculation:", 10, 10,

"(%s / %s + ln(3 \* %s / 2))", 10, "---------------------------- = ERROR because division by 0 is not allowed", 10, " ( %s - %s + 1 )", 10, 10, 10, 0

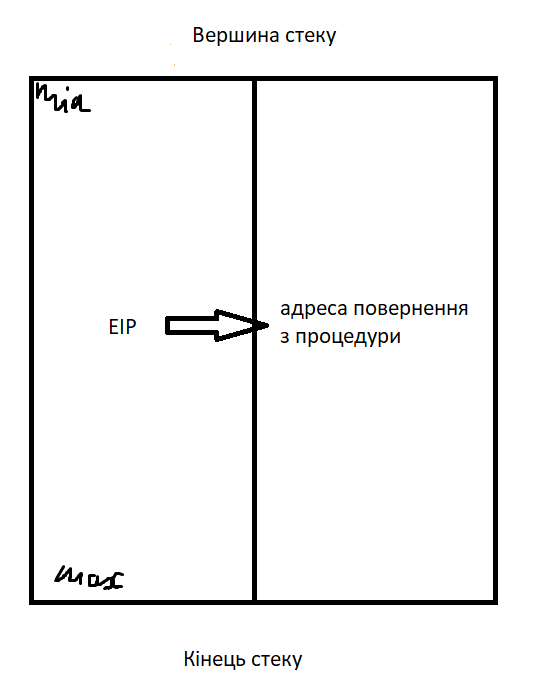
NESTER\_INCORRECT\_AREA\_FORMAT db "Lab Variant: 16", 10, 10, 10,

"Formula:", 10, 10, "(c / d + ln(3 \* a / 2))", 10, "-------------------------,", " where: a = %s, b = %s, c = %s, d = %s", 10, " (c - b + 1)", 10, 10, 10,

"Calculation:", 10, 10,

"(%s / %s + ln(3 \* %s / 2))", 10, "---------------------------- = ERROR, num under logarifm must be > 0", 10, " ( %s - %s + 1 )", 10, 10, 10, 0

.CODE

NESTER\_FIRST\_PROC PROC --------> 

FLD QWORD PTR [ecx + esi\*8] ;c

FLD QWORD PTR[edx + esi\*8] ; d

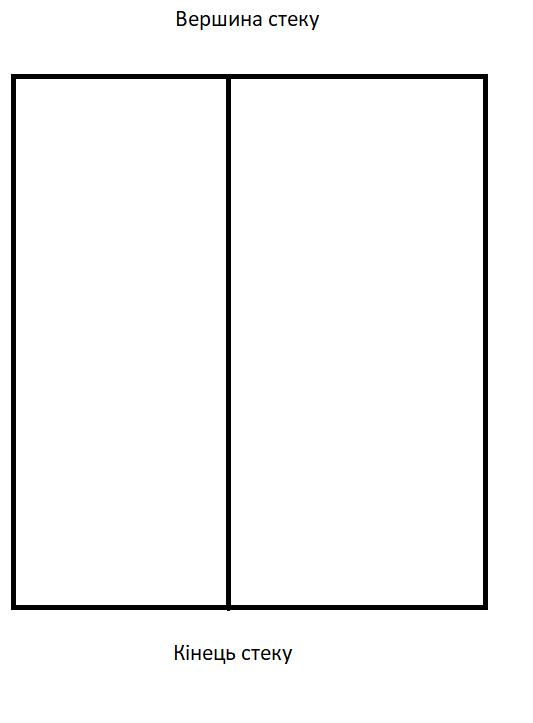
FTST

FNSTSW ax

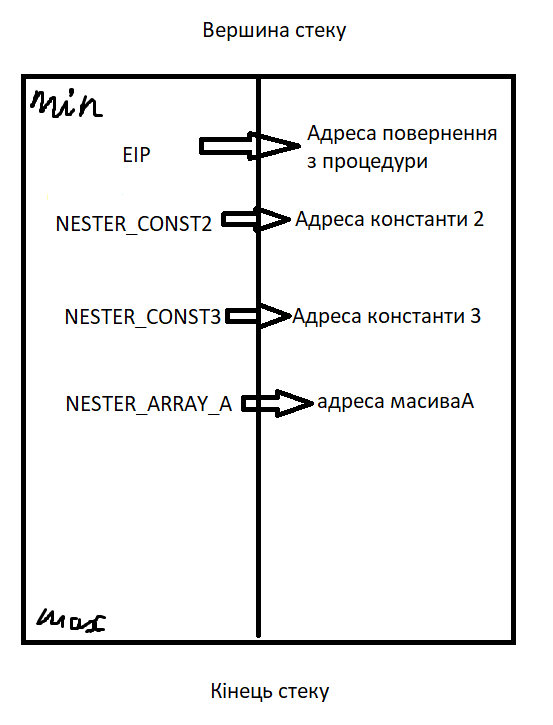
SAHF

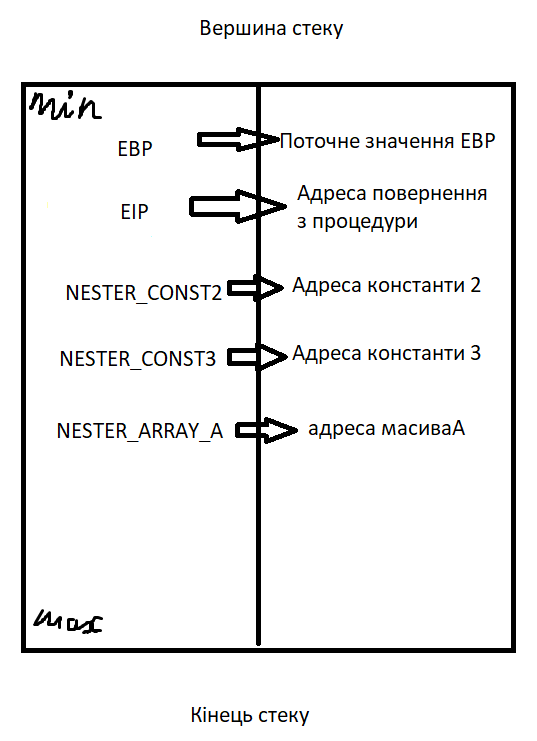
JZ NESTER\_ZERO\_DENOMINATOR\_ERROR ; checking for zero denominator for c/d expression

FDIV ; dividing c on d

RET ------------>

NESTER\_FIRST\_PROC ENDP

NESTER\_SECOND\_PROC PROC --------> 

PUSH ebp ----------> 

MOV ebp, esp

MOV eax, [ebp+16]

MOV ebx, [ebp+12]

MOV ecx, [ebp+8]

FLD QWORD PTR [eax + esi \* 8] ;a

FTST

FSTSW AX

SAHF

JBE NESTER\_INCORRECT\_AREA ; checking if a <= 0

FMUL QWORD PTR [ebx] ;multiplying 3 and a

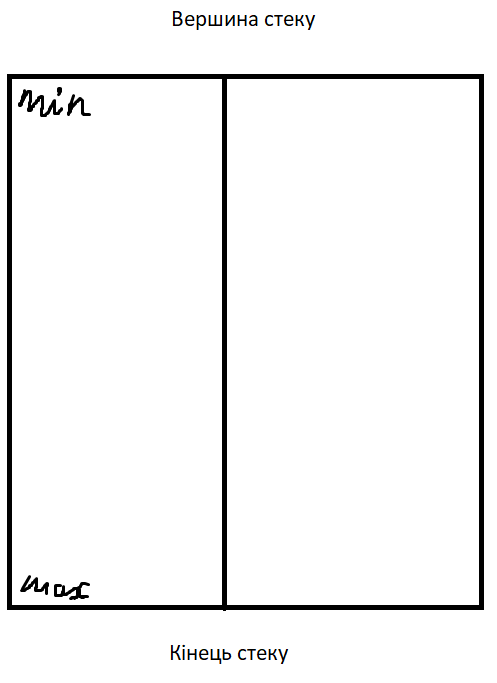
FDIV QWORD PTR [ecx] ;dividing 3a on 2

FLDln2

FXCH st(1)

FYL2X ;ln(3a/2)

POP ebp -----------> 

RET 12 --------------> 

NESTER\_SECOND\_PROC ENDP

NESTER\_TO\_STR MACRO

INVOKE FloatToStr2, NESTER\_ARRAY\_A[esi\*8], addr NESTER\_TEXT\_ARRAY\_A[esi\*8]

INVOKE FloatToStr2, NESTER\_ARRAY\_B[esi\*8], addr NESTER\_TEXT\_ARRAY\_B[esi\*8]

INVOKE FloatToStr2, NESTER\_ARRAY\_C[esi\*8], addr NESTER\_TEXT\_ARRAY\_C[esi\*8]

INVOKE FloatToStr2, NESTER\_ARRAY\_D[esi\*8], addr NESTER\_TEXT\_ARRAY\_D[esi\*8]

INVOKE FloatToStr2, NESTER\_RESULT, addr NESTER\_TEXT\_RESULT

ENDM

NESTER\_CREATE\_MESSAGE MACRO

NESTER\_TO\_STR

INVOKE wsprintf, addr NESTER\_MESSAGE, addr NESTER\_MESSAGE\_FORMAT,

addr NESTER\_TEXT\_ARRAY\_A[esi \* 8], addr NESTER\_TEXT\_ARRAY\_B[esi \* 8], addr NESTER\_TEXT\_ARRAY\_C[esi \* 8], addr NESTER\_TEXT\_ARRAY\_D[esi \* 8],

addr NESTER\_TEXT\_ARRAY\_C[esi \* 8], addr NESTER\_TEXT\_ARRAY\_D[esi \* 8], addr NESTER\_TEXT\_ARRAY\_A[esi \* 8],

addr NESTER\_TEXT\_RESULT,addr NESTER\_TEXT\_ARRAY\_C[esi \* 8], addr NESTER\_TEXT\_ARRAY\_B[esi \* 8]

ENDM

NESTER\_CREATE\_ERROR\_MESSAGE MACRO param

NESTER\_TO\_STR

INVOKE wsprintf, addr NESTER\_MESSAGE, param ,

addr NESTER\_TEXT\_ARRAY\_A[esi \* 8], addr NESTER\_TEXT\_ARRAY\_B[esi \* 8], addr NESTER\_TEXT\_ARRAY\_C[esi \* 8], addr NESTER\_TEXT\_ARRAY\_D[esi \* 8],

addr NESTER\_TEXT\_ARRAY\_C[esi \* 8], addr NESTER\_TEXT\_ARRAY\_D[esi \* 8], addr NESTER\_TEXT\_ARRAY\_A[esi \* 8],

addr NESTER\_TEXT\_ARRAY\_C[esi \* 8], addr NESTER\_TEXT\_ARRAY\_B[esi \* 8]

ENDM

NESTER\_CREATE\_ZERO\_DENOMINATOR\_ERROR\_MESSAGE MACRO

NESTER\_CREATE\_ERROR\_MESSAGE addr NESTER\_ZERO\_DENOMITATOR\_FORMAT

ENDM

NESTER\_CREATE\_INVALID\_AREA\_ERROR\_MESSAGE MACRO

NESTER\_CREATE\_ERROR\_MESSAGE addr NESTER\_INCORRECT\_AREA\_FORMAT,

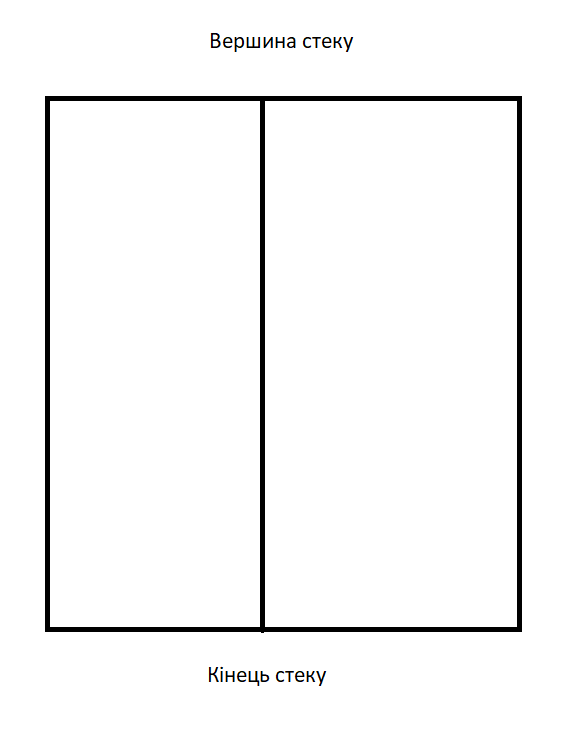
ENDM

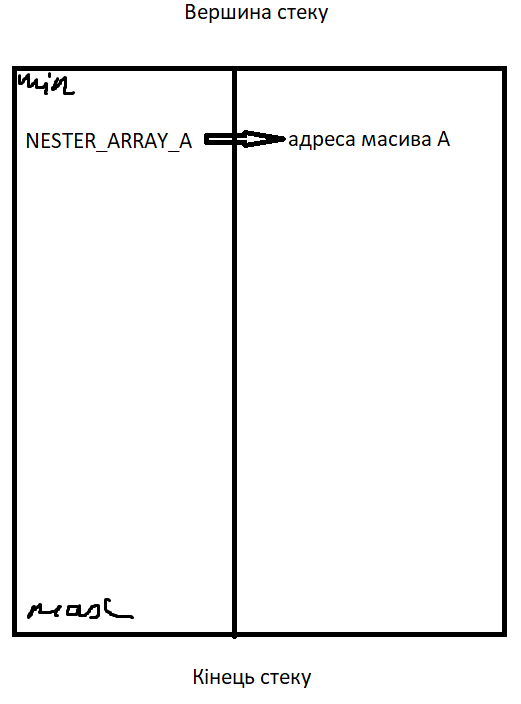
NESTER\_CALCULATE MACRO

FINIT

MOV ecx, OFFSET NESTER\_ARRAY\_C

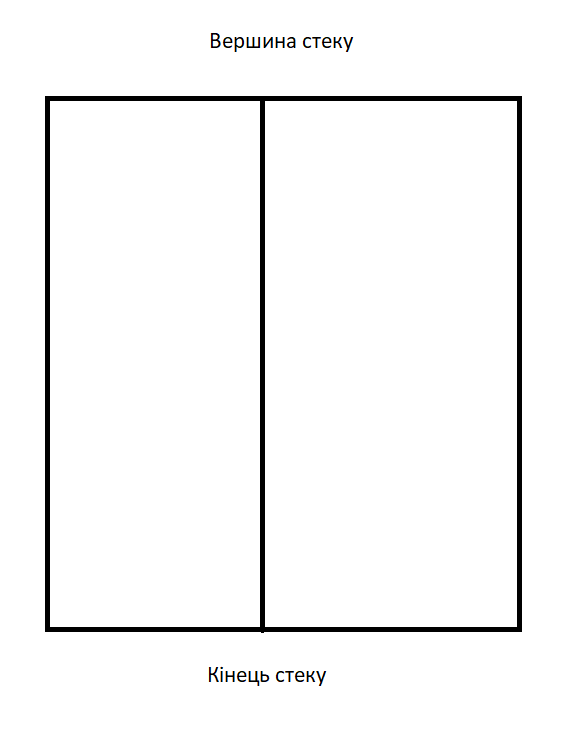
MOV edx, OFFSET NESTER\_ARRAY\_D

CALL NESTER\_FIRST\_PROC; --------->

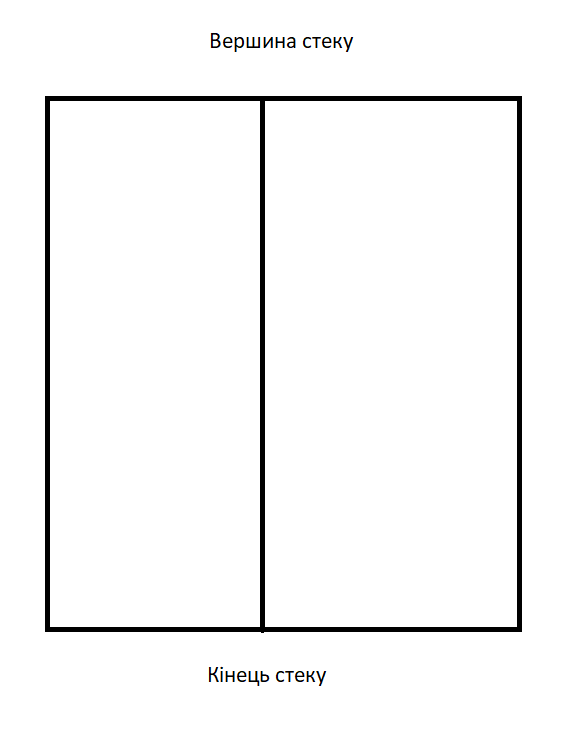
PUSH OFFSET NESTER\_ARRAY\_A --------> 

PUSH OFFSET NESTER\_CONST3 --------> 

PUSH OFFSET NESTER\_CONST2 -------> 

CALL NESTER\_SECOND\_PROC ---------> 

FADD ;adding c/d to ln(3a/2)

CALL NESTER\_THIRD\_PROC --------> 

FTST

FNSTSW ax

SAHF

JZ NESTER\_ZERO\_DENOMINATOR\_ERROR ;checking if denominator c-b+1 = 0

FDIV ; final dividing

FSTP NESTER\_RESULT ;result will be (ln(3a/2) + c/d)/(c-b+1)

NESTER\_CREATE\_MESSAGE

INVOKE MessageBox, 0, addr NESTER\_MESSAGE, addr NESTER\_WINDOW\_TITLE, 0

JMP NESTER\_SKIP\_ERRORS

NESTER\_ZERO\_DENOMINATOR\_ERROR:

NESTER\_CREATE\_ZERO\_DENOMINATOR\_ERROR\_MESSAGE

INVOKE MessageBox, 0, addr NESTER\_MESSAGE, addr NESTER\_WINDOW\_TITLE, 0

JMP NESTER\_SKIP\_ERRORS

NESTER\_INCORRECT\_AREA:

NESTER\_CREATE\_INVALID\_AREA\_ERROR\_MESSAGE

INVOKE MessageBox, 0, addr NESTER\_MESSAGE, addr NESTER\_WINDOW\_TITLE, 0

NESTER\_SKIP\_ERRORS:

MOV NESTER\_MESSAGE, 0h

ENDM

NESTER\_SIXTH\_LAB:

MOV esi, 0

.WHILE esi < 6

NESTER\_CALCULATE

ADD esi, 1

.ENDW

INVOKE ExitProcess, NULL

END NESTER\_SIXTH\_LAB

**Код мого модулю**

.386

.MODEL flat, stdcall

OPTION casemap :none

public NESTER\_THIRD\_PROC

extern NESTER\_ARRAY\_C:qword, NESTER\_ARRAY\_B:qword, NESTER\_CONST1:qword

.code

NESTER\_THIRD\_PROC proc

FLD NESTER\_ARRAY\_C[esi\*8] ;c

FSUB NESTER\_ARRAY\_B[esi\*8] ; subtractiob b from c

FADD NESTER\_CONST1 ; adding 1 to c-b

RET

NESTER\_THIRD\_PROC endp

end

**Висновки**

Використання процедур підвищує модульність програми. При виконанні лабораторної роботи я використовував процедури з різними способами передачі параметрів, а саме: через регістри, стек, та за допомогою директив EXTERN і PUBLIC.

Порівнюючи ці способи маю сказати, що передача параметрів через регістри, наприклад EAX, EBX, ECX и EDX більш ефективний, бо звернення до регістрів виконується швидше ніж операції з пам’яттю. Але ми обмежені кількістю доступних регістрів, тому цей метод не завжди є доцільним для використання. У випадках коли кількість параметрів більша за кількість цих регістрів, або нам необхідна динамічна кількість параметрів він нам точно не підійде.

Передача параметрів через стек більш розповсюджений спосіб, що не дивно. Поняття стеку мені було відоме ще зі школи, де ми вивчали с++. А також постійно використовувалося на інших предметах таких як алгоритми і структури даних. При передачі параметрів через стек значення кладеться у вершину стеку перед викликом функції, яка в свою чергу при виклику дістає значення зі стека. Сам стек це структура, що працює за принципом FIFO, First In, First Out. Параметри звичайно кладуть у стек у зворотньому порядку. Цей спосіб використовують коли нам необхідно використати більше параметрів, чим є доступних регістрів. Метод доволі простий у використанні та розумінні, а ще широко розповсюджений, як і наступний.

Використання директив EXTERN і PUBLIC дозволяють писати код, що легко маштабується, що є безперечно плюсом. Директива EXTERN каже, що символ визначений в іншому модулі і може бути використаний у цьому, а PUBLIC навпаки вказує на те, що символ визначений у цьому модулі, але може бути використаний в інших.

Залежно від ситуації треба правильно обирати конкретний метод передачі параметрів щоб підвищити продуктивність програми. У такій маленькій програмі результат оптимального вибору буде побачити важко, але на більш об’ємних він буде значним.