**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 №6**

з дисципліни  
 «Системне програмування»

на тему

“Обчислення арифметичних виразів і трансцендентних функцій. Використання команд співпроцесора ix87”

Виконала: Перевірив:

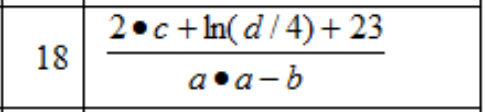
Студентка 2-го курсу групи ІМ-21 Павлов Валерій Георгійович

Рабійчук Дар’я Олександрівна  
номер у списку групи: 18

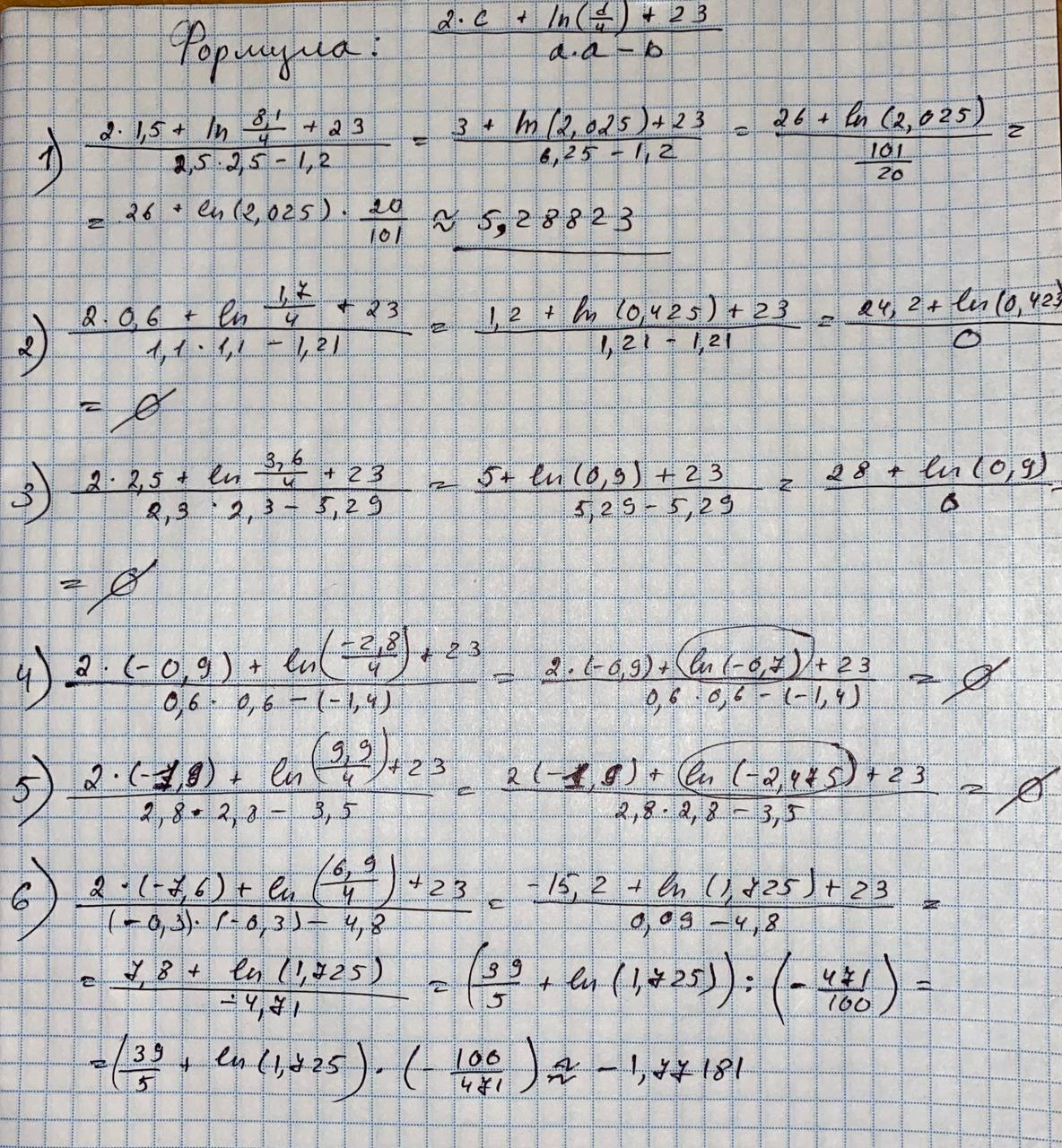
**Київ 2024**

**Мета роботи:** Вивчення команд Асемблера для арифметики з плаваючою комою і здобуття навичок виконання розрахунків з елементами масивів.

**Завдання за варіантом:**

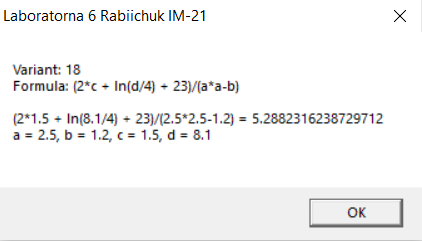
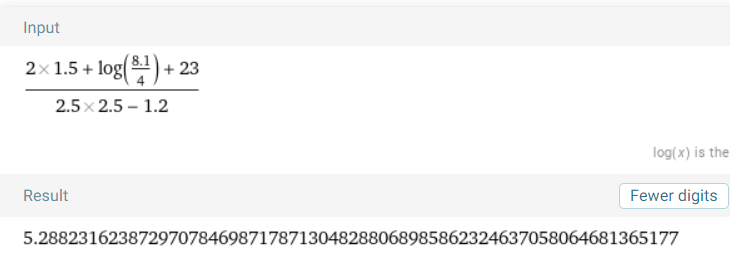
****

**Розрахунки:**

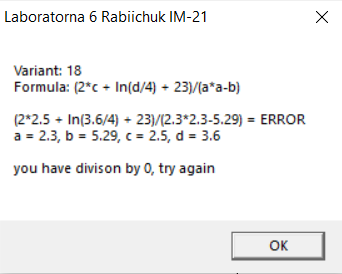


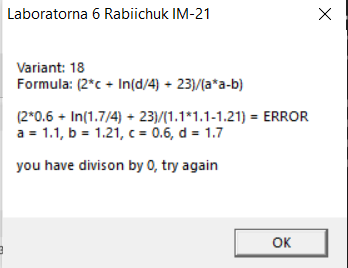
**Скриншоти виконання програми:**

Я зробила 6 прикладів, 2 з яких показують точність розрахунків, 2 повідомлення про помилку при діленні на 0 та 2 про недопустимих значеннях логарифма.

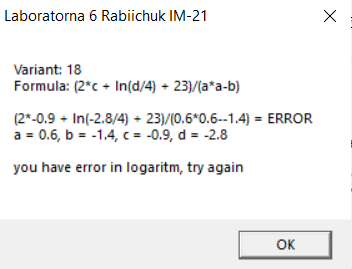
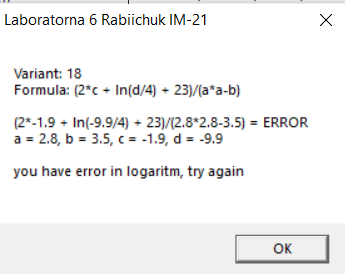
****

Перший приклад демонструє правильність виконання програми. На другому скриншоті додала розрахунки з Wolfram калькулятора для порівняння. Бачимо, що відповідь співпадає до 15 знаку після коми.

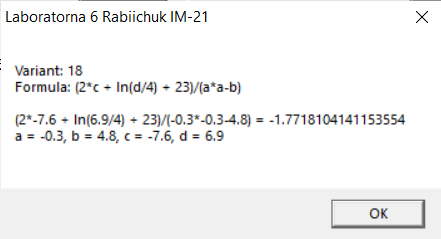
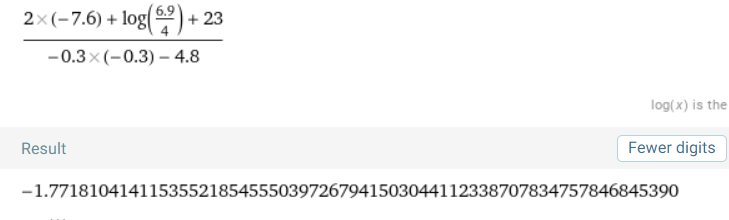




На другому та третьому прикладі бачимо ділення на 0 та відповідне повідомлення в діалоговому вікні.

На четвертому та п'ятому прикладах продемонструвала повідомлення програми на недопустиме для натурального логарифма значення.



В останньому шостому прикладі також бачимо досить точні розрахунки порівнюючи з результатом Wolfram калькулятора.

**Лістинг програми:**

.386

.model flat, stdcall

option casemap: none

include \masm32\include\masm32rt.inc

.data?

myResultOfCalculation dq 1 dup(?)

firstPartOfFormula dt 16 dup(?)

secondPartOfFormula dt 16 dup(?)

myFinalResultOfCalculation dt 64 dup(?)

myNumberForFormulaA db 32 dup(?)

myNumberForFormulaB db 32 dup(?)

myNumberForFormulaC db 32 dup(?)

myNumberForFormulaD db 32 dup(?)

myMessageRabiichukLab6 db 128 dup(?)

labMyMessageBuffer6 db 128 dup(?)

.data

labTitle db "Laboratorna 6 Rabiichuk IM-21", 0

myMessageNormalText db "Variant: 18", 10,

"Formula: (2\*c + ln(d/4) + 23)/(a\*a-b)", 10, 10,

"(2\*%s + ln(%s/4) + 23)/(%s\*%s-%s) = %s", 10,

"a = %s, b = %s, c = %s, d = %s", 10, 10, 0

myMessageErrorText1 db "Variant: 18", 10,

"Formula: (2\*c + ln(d/4) + 23)/(a\*a-b)", 10, 10,

"(2\*%s + ln(%s/4) + 23)/(%s\*%s-%s) = ERROR", 10,

"a = %s, b = %s, c = %s, d = %s", 10, 10,

"you have divison by 0, try again", 10, 10, 0

myMessageErrorText2 db "Variant: 18", 10,

"Formula: (2\*c + ln(d/4) + 23)/(a\*a-b)", 10, 10,

"(2\*%s + ln(%s/4) + 23)/(%s\*%s-%s) = ERROR", 10,

"a = %s, b = %s, c = %s, d = %s", 10, 10,

"you have error in logaritm, try again", 10, 10, 0

arrayForCalculationA dq 2.5, 1.1, 2.3, 0.6, 2.8, -0.3

arrayForCalculationB dq 1.2, 1.21, 5.29, -1.4, 3.5, 4.8

arrayForCalculationC dq 1.5, 0.6, 2.5, -0.9, -1.9, -7.6

arrayForCalculationD dq 8.1, 1.7, 3.6, -2.8, -9.9, 6.9

two dq 2.0

four dq 4.0

twentyThree dq 23.0

displayMessage macro windowMessage, windowTitle

invoke MessageBox, 0, offset windowMessage, offset windowTitle, 0

endm

.code

myProgram6Laboratorna:

mov ebp, 0

.while ebp < 6

invoke FloatToStr2, arrayForCalculationA[ebp \* 8], addr myNumberForFormulaA

invoke FloatToStr2, arrayForCalculationB[ebp \* 8], addr myNumberForFormulaB

invoke FloatToStr2, arrayForCalculationC[ebp \* 8], addr myNumberForFormulaC

invoke FloatToStr2, arrayForCalculationD[ebp \* 8], addr myNumberForFormulaD

; Ініціалізація FPU

finit

; Загрузка даних

fld arrayForCalculationD[ebp \* 8] ; Завантаження d

fld four ; Завантаження числа 4

fdiv ; Обчислення d / 4

fldln2 ; Обчислення ln(d/4)

fxch ; Обмін верхніми двома значеннями в стеку

fyl2x ; Обчислення результату ln(d/4)

ftst

fnstsw ax

sahf

jz errorValidValues

; Завантаження і обчислення першої частини формули

fld st(0) ; Копіювання результату ln(d/4) для подальших обчислень

fld arrayForCalculationC[ebp \* 8] ; Завантаження c

fld two ; Завантаження числа 2

fmul ; Обчислення 2\*c

fadd ; Обчислення 2\*c + ln(d/4)

fld twentyThree ; Завантаження числа 23

fadd ; Обчислення 2\*c + ln(d/4) + 23

fstp qword ptr [firstPartOfFormula] ; Збереження першої частини формули в long double

; Обчислення знаменника (a^2 - b)

fld arrayForCalculationA[ebp \* 8] ; Загрузка a

fmul st(0), st(0) ; Обчислення a^2

fld arrayForCalculationB[ebp \* 8] ; Загрузка b

fsub ; Обчислнння a^2 - b

fstp qword ptr [secondPartOfFormula] ; Збереження другої частини виразу в qword ptr

; Перевірка флагів статусу FPU

ftst

fnstsw ax

sahf

jbe errorFormulaWIthZero ; Перехід до обробника помилки, якщо ділення на нуль

fld qword ptr [firstPartOfFormula] ; st(1)

fdiv qword ptr [secondPartOfFormula] ; Ділення чисельник на знаменник

fstp qword ptr [myResultOfCalculation] ; Збереження результату в qword ptr

jmp formulaHaveNoErrors

; Підготовка строкового представлення для виводу повідомлення

formulaHaveNoErrors:

invoke FloatToStr2, myResultOfCalculation, addr myFinalResultOfCalculation

invoke wsprintf, addr labMyMessageBuffer6, addr myMessageNormalText,

addr myNumberForFormulaC, addr myNumberForFormulaD,

addr myNumberForFormulaA, addr myNumberForFormulaA, addr myNumberForFormulaB,

addr myFinalResultOfCalculation,

addr myNumberForFormulaA, addr myNumberForFormulaB, addr myNumberForFormulaC, addr myNumberForFormulaD

invoke szCatStr, addr myMessageRabiichukLab6, addr labMyMessageBuffer6

displayMessage myMessageRabiichukLab6, labTitle

jmp nextIterationOfFormula

errorFormulaWIthZero:

invoke wsprintf, addr labMyMessageBuffer6, addr myMessageErrorText1,

addr myNumberForFormulaC, addr myNumberForFormulaD,

addr myNumberForFormulaA, addr myNumberForFormulaA, addr myNumberForFormulaB,

addr myNumberForFormulaA, addr myNumberForFormulaB, addr myNumberForFormulaC, addr myNumberForFormulaD

invoke szCatStr, addr myMessageRabiichukLab6, addr labMyMessageBuffer6

displayMessage myMessageRabiichukLab6, labTitle

jmp nextIterationOfFormula

errorValidValues:

invoke wsprintf, addr labMyMessageBuffer6, addr myMessageErrorText2,

addr myNumberForFormulaC, addr myNumberForFormulaD,

addr myNumberForFormulaA, addr myNumberForFormulaA, addr myNumberForFormulaB,

addr myNumberForFormulaA, addr myNumberForFormulaB, addr myNumberForFormulaC, addr myNumberForFormulaD

invoke szCatStr, addr myMessageRabiichukLab6, addr labMyMessageBuffer6

displayMessage myMessageRabiichukLab6, labTitle

jmp nextIterationOfFormula

nextIterationOfFormula:

mov myMessageRabiichukLab6, 0h

inc ebp

.endw

invoke ExitProcess, 0

end myProgram6Laboratorna

**Висновок:**

У результаті виконання даної лабораторної роботи я навчилася працювати з новими для мене командами, які використовуються для розрахунків чисел з плаваючою комою, наприклад: FDIV, FMUL, FSUB, FADD, FYL2X. У процесі написання програми я намагалася обробити усі можливі випадки обчислень свого виразу. При виявлені неможливого за ОДЗ значенням натурального логарифму моя програму виводить відповідне повідомлення про помилку, при обробці наявності нуля при діленні також можна побачити відповідний текст з помилкою, а при обробці значень масиву, які відповідають умовам ОДЗ ми можемо побачити у діалоговому вікні відповідь. Я порахувала свої приклади і вручну і за допомогою Wolfram калькулятора та при порівняні побачила, що точність розрахунків моєї програми складає 0.00000000000001, що є досить високою точністю.