**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 курсу ФІОТ доц. Павлов В. Г.

групи ІМ-22

Довженко Антон Андрійович

номер в списку групи: 9

Київ 2024

**Мета роботи:** Вивчення прийомів модульного програмування, методів звернення до процедур і передачі в них параметрів.

**Завдання:**  
Номер варіанту – 9, бо в групі в списку я під номером 9:  
(2 \* c - d / 23) / ln(b - a / 4)

**Контрольні приклади:**

*1. Чисельник та знаменник більші за нуль.*

Візьмемо такі значення:

a = 3.4, b = 9.3, c = 12.6, d = 18.1

(2 \* c - d / 23) / ln(b - a / 4) = (2 \* 12.6 – 18.1 / 23) / ln (9.3 – 3.4 / 4) = (1123 / 46) / ln (8,45) = 24.41304347826 / 2.13416644137 = 11.43914692173

*2. Чисельник більший за нуль, а знаменник менший.*

a = 10.6, b = 3.5, c = 2.7, d = 1.5

(2 \* c - d / 23) / ln(b - a / 4) = (2 \* 2.7 – 1.5 / 23) / ln(3.5 – 10.6 / 4) = (5.4 – 3 / 2 / 23) / ln(3,5 – 2,65) = (27/5 – 3/46) / ln(0.85) = (1227/230)/−0.16251892950 = 5.33478260870/-0.16251892950 = −32.82560760837

*3. Чисельник менший за нуль, а знаменник більший.*

a = 11.3, b = 19.7, c = 1.3, d = 71.2

(2 \* c - d / 23) / ln(b - a / 4) = (2 \* 1.3 – 71.2/23) / ln(19.7-11.3/4) = (2.6 – 356 / 5 / 23) / ln(19.7 – 2.85) = (13/5 – 356/115) / ln(16.85) = -0.49565217391 / 2.82583323676 = -0,17540036244

4. *Знаменник дорівнює нулю (ділення на нуль).*

a = 10.4 , b = 3.6, c = 2.8, d = 4.6

(2 \* c - d / 23) / ln(b - a / 4) = (2 \* 2.8 - 4.6 / 23) / ln(3.6 – 10.4 / 4) =   
(5.6 – 4.6/23) / ln (3.6 – 2.6) = (5.6 – 4.6/23) / ln (1) = (5.6 – 4.6/23) / 0

*5. Чисельник та знаменник менші за нуль.*

a = 88.8, b = 22.6, c = 2.6, d = 213.3

(2 \* c - d / 23) / ln(b - a / 4) = (2 \* 2.6 – 213.3 / 23) / ln(22.6 – 88.8 / 4) = (5,2 – 2133 / 10 / 23) / ln(22,6 – 22,2) = (-937/230) / ln (0,4) = -4.07391304348 /  
-0.91629073187 = 4.44609216462

Тепер додаткові контрольні приклади:

*6. Вираз під натуральним логарифмом є меншим за нуль, що не може бути:*

a = 16.2 , b = 3.9, c = 6.4, d = 17.3

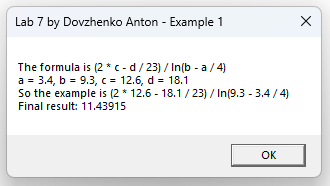
(2 \* c - d / 23) / ln(b - a / 4) = (2 \* 6.4 – 17.3/23) / ln(3.9 – 4.05) = (2 \* 6.4 – 17.3/23) / ln (-0.15)

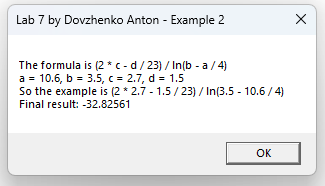
*7. Вираз під натуральним логарифмом дорівнює нулю*

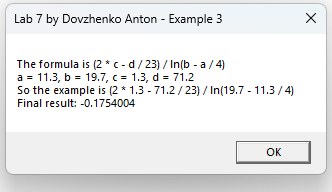
a = 12.4 , b = 3.1, c = 19.1, d = 169.6

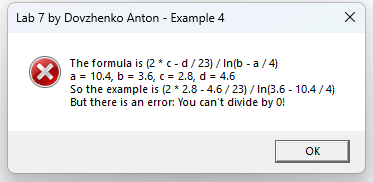
(2 \* c - d / 23) / ln(b - a / 4) = (2 \* 19.1 – 169.6 / 23) / ln(3.1 – 12.4 / 4) =   
(2 \* 19.1 – 169.6 / 23) / ln(3.1 – 3.1) = (2 \* 19.1 – 169.6 / 23) / ln(0)

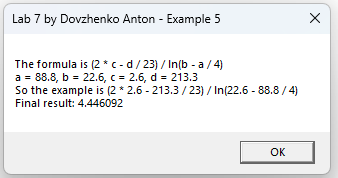
**Демонстрація роботи програми**

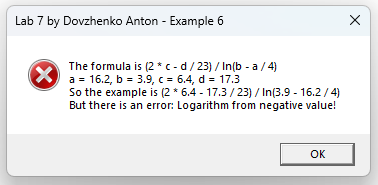


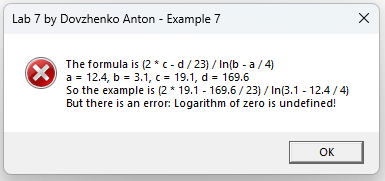












Примітка: Замість 5 звичних контрольних прикладів, було зроблено 7 (аналогічно і в програмі записано 7 наборів чисел, бо минулого разу за це була знижена оцінка), щоб охопити всі порушення області визначення та перевірити правильність класифікації та виводу різних помилок програмою.

Бачимо, що програма обраховує приклади коректно. Також варто зазначити, що програма правильно виявляє та класифікує помилки, виводячи повідомлення про помилку з номером прикладу, та його загальним видом по формулі.

**Лістинг програми**

*7-9-IM-22-Dovzhenko.asm*

.386

.model flat, stdcall

option casemap:none

include \masm32\include\masm32rt.inc

public DovzhenkoCurrentA, DovzhenkoCurrentB, DovzhenkoFour, DovzhenkoDenominatorVal, DovzhenkoZero

extern DovzhenkoGetDenominator:proto

.data

DovzhenkoA dq 3.4, 10.6, 11.3, 10.4, 88.8, 16.2, 12.4

DovzhenkoB dq 9.3, 3.5, 19.7, 3.6, 22.6, 3.9, 3.1

DovzhenkoC dq 12.6, 2.7, 1.3, 2.8, 2.6, 6.4, 19.1

DovzhenkoD dq 18.1, 1.5, 71.2, 4.6, 213.3, 17.3, 169.6

DovzhenkoZero dq 0.0

DovzhenkoTwo dq 2.0

DovzhenkoFour dq 4.0

DovzhenkoTwentyThree dq 23.0

DovzhenkoNumeratorPart1 dq 0.0

DovzhenkoNumeratorPart2 dq 0.0

DovzhenkoNumeratorVal dt 0.0

DovzhenkoDenominatorVal dt 0.0

DovzhenkoResultrVal dq 0.0

LogErrorIndicator dq -9999.0

LogZeroIndicator dq -8888.0

DovzhenkoCurrentA dq 0.0

DovzhenkoCurrentB dq 0.0

DovzhenkoTitleTemplate db "Lab 7 by Dovzhenko Anton - Example %d", 0

DovzhenkoExampleTitle db 64 dup(0)

DovzhenkoResultsWin db "The formula is (2 \* c - d / 23) / ln(b - a / 4)", 13,

"a = %s, b = %s, c = %s, d = %s", 13,

"So the example is (2 \* %s - %s / 23) / ln(%s - %s / 4)", 13,

"Final result: %s", 0

DovzhenkoErrorExample db "The formula is (2 \* c - d / 23) / ln(b - a / 4)", 13,

"a = %s, b = %s, c = %s, d = %s", 13,

"So the example is (2 \* %s - %s / 23) / ln(%s - %s / 4)", 13,

"But there is an error: %s", 0

DovzhenkoZeroDenomErrorMsg db "You can't divide by 0!", 0

DovzhenkoNegLogErrorMsg db "Logarithm from negative value!", 0

DovzhenkoLogZeroErrorMsg db "Logarithm of zero is undefined!", 0

DovzhenkoAStr db 64 dup(0)

DovzhenkoBStr db 64 dup(0)

DovzhenkoCStr db 64 dup(0)

DovzhenkoDStr db 64 dup(0)

DovzhenkoResultString db 64 dup(0)

DovzhenkoDisplayBuff db 256 dup(0)

.code

; (2 \* c - d / 23)

; First part (2 \* c) with parameter passing through eax

CalcTwoC proc

fld qword ptr [eax]

fmul qword ptr [ebx]

ret

CalcTwoC endp

; Second part (- d / 23), through the stack

CalcNegDiv proc

pop ebx

pop ecx

fld qword ptr [ecx]

pop ecx

fdiv qword ptr [ecx]

fchs

push ebx

ret

CalcNegDiv endp

main proc

mov edi, 0

DovzhenkoCalculateLoop:

cmp edi, 7

je DovzhenkoEndProgram

mov eax, edi

inc eax

invoke wsprintf, addr DovzhenkoExampleTitle, addr DovzhenkoTitleTemplate, eax

finit

; (2 \* c)

lea eax, [DovzhenkoC + edi \* 8]

lea ebx, [DovzhenkoTwo]

call CalcTwoC

fstp qword ptr [DovzhenkoNumeratorPart1]

; (- d / 23)

push offset DovzhenkoTwentyThree ; Load 23 through the stack

lea ecx, [DovzhenkoD + edi \* 8] ; Load d through the stack

push ecx

call CalcNegDiv

pop ecx

fstp qword ptr [DovzhenkoNumeratorPart2]

fld [DovzhenkoNumeratorPart1]

fadd [DovzhenkoNumeratorPart2]

fstp [DovzhenkoNumeratorVal]

fld [DovzhenkoA + edi \* 8]

fstp DovzhenkoCurrentA

fld [DovzhenkoB + edi \* 8]

fstp DovzhenkoCurrentB

invoke FloatToStr, [DovzhenkoA + edi \* 8], addr DovzhenkoAStr

invoke FloatToStr, [DovzhenkoB + edi \* 8], addr DovzhenkoBStr

invoke FloatToStr, [DovzhenkoC + edi \* 8], addr DovzhenkoCStr

invoke FloatToStr, [DovzhenkoD + edi \* 8], addr DovzhenkoDStr

call DovzhenkoGetDenominator

fld [DovzhenkoDenominatorVal]

fcomp qword ptr [LogErrorIndicator]

fstsw ax

sahf

je DovzhenkoNegLogError

fld [DovzhenkoDenominatorVal]

fcomp qword ptr [LogZeroIndicator]

fstsw ax

sahf

je DovzhenkoLogZeroError

fld [DovzhenkoDenominatorVal]

fcomp qword ptr [DovzhenkoZero]

fstsw ax

sahf

je DovzhenkoZeroDivideError

fld [DovzhenkoNumeratorVal]

fld [DovzhenkoDenominatorVal]

fdiv

fstp [DovzhenkoResultrVal]

invoke FloatToStr, [DovzhenkoResultrVal], addr DovzhenkoResultString

invoke wsprintf, addr DovzhenkoDisplayBuff, addr DovzhenkoResultsWin,

addr DovzhenkoAStr, addr DovzhenkoBStr, addr DovzhenkoCStr, addr DovzhenkoDStr,

addr DovzhenkoCStr, addr DovzhenkoDStr, addr DovzhenkoBStr, addr DovzhenkoAStr,

addr DovzhenkoResultString

invoke MessageBox, NULL, addr DovzhenkoDisplayBuff, addr DovzhenkoExampleTitle, MB\_OK

inc edi

jmp DovzhenkoCalculateLoop

DovzhenkoNegLogError:

invoke wsprintf, addr DovzhenkoDisplayBuff, addr DovzhenkoErrorExample,

addr DovzhenkoAStr, addr DovzhenkoBStr, addr DovzhenkoCStr, addr DovzhenkoDStr,

addr DovzhenkoCStr, addr DovzhenkoDStr, addr DovzhenkoBStr, addr DovzhenkoAStr,

addr DovzhenkoNegLogErrorMsg

invoke MessageBox, NULL, addr DovzhenkoDisplayBuff, addr DovzhenkoExampleTitle, MB\_ICONERROR or MB\_OK

jmp DovzhenkoNextExample

DovzhenkoLogZeroError:

invoke wsprintf, addr DovzhenkoDisplayBuff, addr DovzhenkoErrorExample,

addr DovzhenkoAStr, addr DovzhenkoBStr, addr DovzhenkoCStr, addr DovzhenkoDStr,

addr DovzhenkoCStr, addr DovzhenkoDStr, addr DovzhenkoBStr, addr DovzhenkoAStr,

addr DovzhenkoLogZeroErrorMsg

invoke MessageBox, NULL, addr DovzhenkoDisplayBuff, addr DovzhenkoExampleTitle, MB\_ICONERROR or MB\_OK

jmp DovzhenkoNextExample

DovzhenkoZeroDivideError:

invoke wsprintf, addr DovzhenkoDisplayBuff, addr DovzhenkoErrorExample,

addr DovzhenkoAStr, addr DovzhenkoBStr, addr DovzhenkoCStr, addr DovzhenkoDStr,

addr DovzhenkoCStr, addr DovzhenkoDStr, addr DovzhenkoBStr, addr DovzhenkoAStr,

addr DovzhenkoZeroDenomErrorMsg

invoke MessageBox, NULL, addr DovzhenkoDisplayBuff, addr DovzhenkoExampleTitle, MB\_ICONERROR or MB\_OK

jmp DovzhenkoNextExample

DovzhenkoNextExample:

inc edi

jmp DovzhenkoCalculateLoop

DovzhenkoEndProgram:

invoke ExitProcess, 0

main endp

end main

*7-9-IM-22-Dovzhenko-PublicExtern.asm*

.386

.model flat, stdcall

option casemap:none

public DovzhenkoGetDenominator

extern DovzhenkoCurrentA:qword, DovzhenkoCurrentB:qword, DovzhenkoFour:qword, DovzhenkoDenominatorVal:tbyte, DovzhenkoZero:qword

.data

LogErrorIndicator dq -9999.0

LogZeroIndicator dq -8888.0

.code

DovzhenkoGetDenominator proc

; Calculation of (b - a / 4)

fld DovzhenkoCurrentB

fld DovzhenkoCurrentA

fdiv DovzhenkoFour

fsub

; Value under ln check

fld st(0)

fcomp qword ptr [DovzhenkoZero]

fstsw ax

sahf

jb LogNegativeError

je LogZeroError

; ln calculation

fldln2

fld st(1)

fyl2x

fstp [DovzhenkoDenominatorVal]

ret

LogNegativeError:

; return indicator of ln from negative val

fld qword ptr [LogErrorIndicator]

fstp [DovzhenkoDenominatorVal]

ret

LogZeroError:

; return indicator of ln from zero

fld qword ptr [LogZeroIndicator]

fstp [DovzhenkoDenominatorVal]

ret

DovzhenkoGetDenominator endp

end

*7-9-IM-22-Dovzhenko.bat*

@echo off

ml /c /coff "7-9-IM-22-Dovzhenko-PublicExtern.asm"

ml /c /coff "7-9-IM-22-Dovzhenko.asm"

link /subsystem:windows "7-9-IM-22-Dovzhenko" "7-9-IM-22-Dovzhenko-PublicExtern.obj"

7-9-IM-22-Dovzhenko.exe

**Стани стеку (для першого та другого способів)**

|  |  |
| --- | --- |
| Стан 1: Виклик **CalcTwoC** | |
| max |  |
| ~~ |  |
|  | Адреса повернення з **CalcTwoC** в головну програму |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 2: В кінці виконання процедури **CalcTwoC** командою ret дістаємо адресу повернення в основу програму | |
| max |  |
| ~~ | ~~ |
|  |  |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 3: push ecx | |
| max |  |
| ~~ |  |
|  | Адреса константи числа 23 |
|  | Адреса елемента масиву DovzhenkoD |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 4: виклик **CalcNegDiv** | |
| max |  |
| ~~ |  |
|  | Адреса константи числа 23 |
|  | Адреса елемента масиву DovzhenkoD |
| EIP | Адреса повернення з **CalcNegDiv** в головну програму |
| min |  |

|  |  |
| --- | --- |
| Стан 5: після **всіх команд pop** | |
| max |  |
| ~~ | ~~ |
|  |  |
|  |  |
| ~~ | ~~ |
| min |  |

|  |  |
| --- | --- |
| Стан 6: після **push ebx** | |
| max |  |
| ~~ | ~~ |
|  |  |
|  | Адреса елемента масиву DovzhenkoD |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 7: В кінці виконання процедури **CalcNegDiv**  командою ret дістаємо адресу повернення в основу програму і стек повернуто в початковий стан. | |
| max |  |
| ~~ | ~~ |
|  |  |
|  |  |
| min |  |

**Висновки**

Отже, після порівняння результатів, що виводить програма та результатів контрольних обрахунків для всіх типів прикладів (з різними знаками чисельника та знаменника, і з різними помилками), можна дійти до висновку, що програма працює коректно та обраховує всі приклади правильно, виявляє та класифікує помилки також коректно. Було враховано всі варіанти, область визначення ln та неможливість ділення на 0. Варто зазначити, що все одно похибка можлива, бо через структуру програми, її виконання здійснюється з меншою точністю ніж під час ручних контрольних обрахунків.

Також зазначу, що під час написання цієї роботи я ознайомився з командами для обрахунків чисел з плаваючою комою, і успішно застосував їх в програмі, з чого можна впевнено сказати, що програма зроблена правильно.

До висновку вище, після виконання 7 лабораторної роботи, слід додати, що я також засвоїв та застосував на практиці різні методи передачі параметрів. У програмі я успішно застосував три різні види передачі параметрів (через регістри, стек та за допомогою extern і public). Також я дослідив стани стеку під час виконання програми.

Тобто, можу констатувати, що я ознайомився та практично застосував у програмі методи та прийоми модульного програмування і різні способи передачі параметрів, як я і зазначив вище.