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 II курсу ФІОТ доц. Павлов В. Г. групи ІМ-33

Козарезов Кирил Олександрович

номер у списку групи: 10

Київ 2025

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

**Порядок виконання роботи**

1. Вивчити методи звернення до процедур і передачі в них параметрів.

2. Для свого варіанту індивідуального завдання до лабораторної роботи 6 розробити програму на мові Асемблер, в якій використовувати три процедури з різними способами передачі параметрів:

• через регістри;

• через стек;

• за допомогою директив EXTRN та PUBLIC.

3. Для цього чисельник дробу зі свого варіанту індивідуального завдання до лабораторної роботи 6 розділити на два доданка, з яких для першого застосувати передачу параметрів і результату через регістри, а для другого – через стек. Для знаменника використовувати метод оголошення загальних змінних директивами public і extern. Виведення результату виконати в основній програмі.

4. Розрахунки (п. 3) повторити в програмі для 5 значень змінних, причому всі вхідні значення задати дійсними числами у вигляді одновимірних масивів.

5. Для перевірки правильності виконання розрахунків і результатів, що виводяться, заздалегідь виконати контрольні розрахунки, які повинні охоплювати різноманітні сполучення вхідних даних, на які програма повинна надавати вірну відповідь. Проміжні і остаточні результати контрольних розрахунків привести в звіті по лабораторній роботі. Точність розрахунків така ж, як і у лаб. роботі 6.

6. Виконати відладку програми шляхом порівняння розрахованих програмою результатів з контрольними прикладами. Лістинг розробленої програми і скріншоти розрахунків по всіх контрольних прикладах привести в звіті по лабораторній роботі.

7. У протоколі по лабораторній роботі для першого і другого способів передачі параметрів поруч з відповідними командами у лістингу відобразити в графічному вигляді стани стека при зверненні до процедур, виконання у них команд та повернення з процедур до основної програми.

8. Зробити висновки по лабораторній роботі.

**Виконання роботи**

Формула для розрахунку за 2 варіантом: (4\*c + d - 1.0) / (b - tan(a/2)).

Контрольні розрахунки

1. Знаменник і чисельник > 0

a = -5,5  
b = 4,8  
c = 2,7

d = -1,8

Обчислюємо:

(4\*2,7- 1,8 - 1) / (4,8 - tg(-5,5/2)) =8 / 4,8+tg(2,75) =  
= 8 /( 24/5)+tg(11/4)= 40/(24+5tg(11/4))= 1,82353550893042

1. Чисельник > 0, а знаменник < 0

a = -2.7

b = 5,6

c = -1,2

d = 2,9

Обчислюємо:

(4\*(-1,2)+ 2,9 - 1) / (5,6- tg(-2,7/2)) =(29/10) / 5,6+tg(1,35) =  
= (29/10) /( 28/5)+tg(27/20)= -(29/56+10tg(27/20)= -0,288407363790067

1. Знаменник і чисельник <0

Вхідні параметри:

A = -1.047

B = -1.5

C = -0.25

D = 0.5

Обчислюємо:

(4\*-0.25+ 0.5-1)/(-1.5-tg(-1.047/2))=(-3/2)/((-3/2)+tg(0,5235))=-3/-3+2tg(1047/2000) = 1,6255203679091277

1. Чисельник <0 та знаменник >0

a = -2,8

b = 1,5

c = -1,5

d = 0,5

(4\*-1.5+ 0.5-1)/(1.5-tg(-2.8/2))= (-6,5) / ((3/2)-tg(1,4)) = -13/ (3+2tg(7/5)) = -0.890669156664613

Оскільки формула містить тригонометричну функцію, що має свою область визначення, а саме tg(x) – множина всіх дійсних чисел, де x != pi/2 + pi\*n, додамо приклад з порушенням цієї області. Також оскільки pi – ірраціональне число, візьмемо його наближене значення з восьма знаками після коми pi = 3,141592

1. Порушення області визначення tg(-pi/2)

a = 3.141593

b = -0,5236

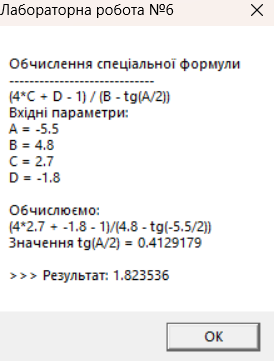
c = 1,5

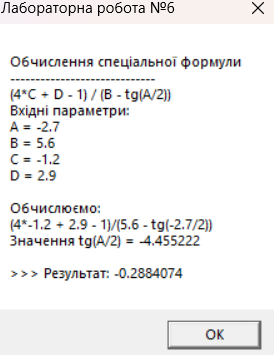
d = 3.1

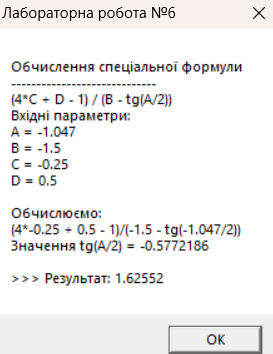
(4\*1.5+3,1-1)/(-0.5236-tg(3.141593/2))= 8.1 /(-0.5236- tg(1.5707965))

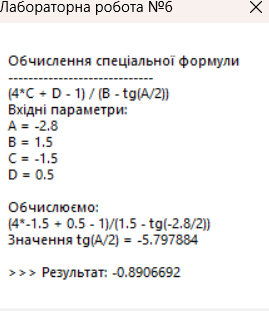
або ж 8.1 / (0.5236- tg(-pi/2)

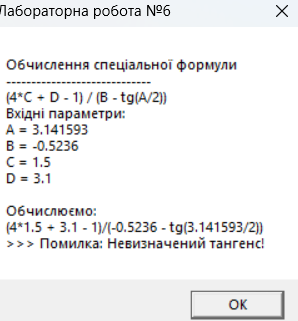
Робота програми











Порівнюючи результати виконання програми із контрольними розрахунками можна сказати, що програма працює коректно. Проте на відміну від контрольних розрахунків, комп’ютер виконує округлення до 6 знаків після коми. Тож перевірена робота програми у таких випадках:

1. Знаменник і чисельник > 0
2. Чисельник < 0, а знаменник > 0
3. Чисельник > 0, а знаменник < 0
4. Знаменник і чисельник < 0
5. Порушення області визначення tg(-pi/2)

Лістинг програми  
(стани стеку виділено червоним кольором)

7-10-IM-33-Kozarezov-PUBLIC.asm

.386

.model flat, stdcall

option casemap:none

include \masm32\include\masm32rt.inc

; Оголошення зовнішніх змінних і міток

EXTRN KozarezovDataA: QWORD

EXTRN KozarezovDataB: QWORD

EXTRN KozarezovTemp: QWORD

EXTRN KozarezovTanVal: QWORD

PUBLIC KozarezovDenom

PUBLIC KozarezovCalcDenom

.data?

KozarezovDenom dq ?

.data

KozarezovTwo dq 2.0

.code

KozarezovCalcDenom PROC C

fld KozarezovDataA[esi\*8]

fdiv KozarezovTwo

fstp KozarezovTemp

fld KozarezovTemp

fptan

fstp st(0)

fstp KozarezovTanVal

fld KozarezovDataB[esi\*8]

fsub KozarezovTanVal

fstp KozarezovDenom

ret

KozarezovCalcDenom ENDP

End

7-10-IM-33-Kozarezov.asm

.386

.model flat, stdcall

option casemap:none

include \masm32\include\masm32rt.inc

; Оголошення зовнішньої змінної та процедури

EXTRN KozarezovDenom: QWORD

EXTRN KozarezovCalcDenom: PROC

; Оголошення змінних і міток як PUBLIC

PUBLIC KozarezovDataA

PUBLIC KozarezovDataB

PUBLIC KozarezovTemp

PUBLIC KozarezovTanVal

PUBLIC KozarezovTanError

.data?

KozarezovOutBuf db 1024 dup(?)

KozarezovTempBuf db 256 dup(?)

KozarezovStrA db 32 dup(?)

KozarezovStrB db 32 dup(?)

KozarezovStrC db 32 dup(?)

KozarezovStrD db 32 dup(?)

KozarezovStrRes db 32 dup(?)

KozarezovStrTanVal db 32 dup(?)

KozarezovTemp dq ?

KozarezovNumer dq ?

KozarezovTanVal dq ?

.data

KozarezovTitle db "Лабораторна робота №7", 0

KozarezovHeader db "Обчислення спеціальної формули", 13, 10

db "-----------------------------", 13, 10, 0

KozarezovFormula db "(4\*C + D - 1) / (B - tg(A/2))", 13, 10, 0

KozarezovInputFmt db "Вхідні параметри:", 13, 10

db "A = %s", 13, 10

db "B = %s", 13, 10

db "C = %s", 13, 10

db "D = %s", 13, 10, 13, 10, 0

KozarezovCalcFmt db "Обчислюємо:", 13, 10

db "(4\*%s + %s - 1)/(%s - tg(%s/2))", 13, 10, 0

KozarezovTanValFmt db "Значення tg(A/2) = %s", 13, 10, 13, 10, 0

KozarezovResultFmt db ">>> Результат: %s", 13, 10, 0

KozarezovErrorDivZero db ">>> Помилка: Ділення на нуль!", 13, 10, 0

KozarezovErrorTanDomain db ">>> Помилка: Невизначений тангенс!", 13, 10, 0

KozarezovDataA dq -5.5, -2.7, -1.047, -2.8, 3.1415926

KozarezovDataB dq 4.8, 5.6, -1.5, 1.5, -0.5236

KozarezovDataC dq 2.7, -1.2, -0.25, -1.5, 1.5

KozarezovDataD dq -1.8, 2.9, 0.5, 0.5, 3.1

KozarezovFour dq 4.0

KozarezovOne dq 1.0

KozarezovTwo dq 2.0

KozarezovPi dq 3.141592

KozarezovPiDiv2 dq 1.570796

KozarezovEpsilon dq 0.000001

.code

; Процедура для обчислення 4\*C через регістри

KozarezovCalcFirst PROC

fld KozarezovDataC[esi\*8] ; Завантажити C

fmul KozarezovFour ; 4\*C

ret

KozarezovCalcFirst ENDP

; Процедура для обчислення D - 1 через стек

KozarezovCalcSecond PROC

push ebp

mov ebp, esp

fld qword ptr [ebp+8] ; Завантажити D зі стека

fsub KozarezovOne ; D - 1

mov esp, ebp

pop ebp

ret 8

KozarezovCalcSecond ENDP

KozarezovDivZero:

invoke wsprintf, addr KozarezovOutBuf, addr KozarezovHeader

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovFormula

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovInputFmt,

addr KozarezovStrA, addr KozarezovStrB,

addr KozarezovStrC, addr KozarezovStrD

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovCalcFmt,

addr KozarezovStrC, addr KozarezovStrD,

addr KozarezovStrB, addr KozarezovStrA

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

invoke FloatToStr, KozarezovTanVal, addr KozarezovStrTanVal

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovTanValFmt, addr KozarezovStrTanVal

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovErrorDivZero

jmp KozarezovShowResult

KozarezovTanError:

invoke wsprintf, addr KozarezovOutBuf, addr KozarezovHeader

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovFormula

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovInputFmt,

addr KozarezovStrA, addr KozarezovStrB,

addr KozarezovStrC, addr KozarezovStrD

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovCalcFmt,

addr KozarezovStrC, addr KozarezovStrD,

addr KozarezovStrB, addr KozarezovStrA

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovErrorTanDomain

jmp KozarezovShowResult

KozarezovBuildOutput:

invoke wsprintf, addr KozarezovOutBuf, addr KozarezovHeader

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovFormula

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovInputFmt,

addr KozarezovStrA, addr KozarezovStrB,

addr KozarezovStrC, addr KozarezovStrD

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovCalcFmt,

addr KozarezovStrC, addr KozarezovStrD,

addr KozarezovStrB, addr KozarezovStrA

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

invoke FloatToStr, KozarezovTanVal, addr KozarezovStrTanVal

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovTanValFmt, addr KozarezovStrTanVal

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

invoke wsprintf, addr KozarezovTempBuf, addr KozarezovResultFmt, addr KozarezovStrRes

invoke szCatStr, addr KozarezovOutBuf, addr KozarezovTempBuf

KozarezovShowResult:

invoke MessageBox, 0, addr KozarezovOutBuf, addr KozarezovTitle, MB\_OK

inc esi

cmp esi, 5

jl KozarezovLoop

invoke ExitProcess, 0

KozarezovMain:

xor esi, esi

KozarezovLoop:

invoke RtlZeroMemory, addr KozarezovOutBuf, sizeof KozarezovOutBuf

invoke RtlZeroMemory, addr KozarezovTempBuf, sizeof KozarezovTempBuf

invoke RtlZeroMemory, addr KozarezovStrRes, sizeof KozarezovStrRes

invoke RtlZeroMemory, addr KozarezovStrTanVal, sizeof KozarezovStrTanVal

finit

invoke FloatToStr, KozarezovDataA[esi\*8], addr KozarezovStrA

invoke FloatToStr, KozarezovDataB[esi\*8], addr KozarezovStrB

invoke FloatToStr, KozarezovDataC[esi\*8], addr KozarezovStrC

invoke FloatToStr, KozarezovDataD[esi\*8], addr KozarezovStrD

; Обчислення чисельника

call KozarezovCalcFirst ; Обчислення 4\*C

fstp KozarezovTemp ; Зберегти перший доданок

push dword ptr KozarezovDataD[esi\*8+4] ; Параметр D у стек

push dword ptr KozarezovDataD[esi\*8]

call KozarezovCalcSecond ; Обчислення D - 1

fadd KozarezovTemp ; Додати до першого доданка

fstp KozarezovNumer ; Зберегти чисельник

; Перевірка області визначення тангенса

fld KozarezovDataA[esi\*8]

fdiv KozarezovTwo

fstp KozarezovTemp

fld KozarezovTemp

fsub KozarezovPiDiv2

fld KozarezovPi

fxch st(1)

fprem

fstp st(1)

fabs

fld KozarezovEpsilon

fxch st(1)

fcomp st(1)

fstsw ax

sahf

fstp st(0)

jbe KozarezovTanError

; Обчислення знаменника

call KozarezovCalcDenom ; Виклик процедури для знаменника

; Перевірка ділення на нуль

fld KozarezovDenom

ftst

fstsw ax

sahf

jz KozarezovDivZero

fstp st(0)

; Обчислення результату

fld KozarezovNumer

fdiv KozarezovDenom

fstp KozarezovTemp

invoke FloatToStr, KozarezovTemp, addr KozarezovStrRes

jmp KozarezovBuildOutput

end KozarezovMain

7-10-IM-33-Kozarezov.bat  
@echo off

ml /c /coff "7-10-IM-33-Kozarezov.asm"

ml /c /coff "7-10-IM-33-Kozarezov-PUBLIC.asm"

polink /subsystem:windows "7-10-IM-33-Kozarezov.obj" "7-10-IM-33-Kozarezov-PUBLIC.obj"

7-10-IM-33-Kozarezov.exe

**Стани стеку**

**Стан 1. Виклик процедури KozarezovCalcFirst**

|  |
| --- |
| max |
| EIP — адреса повернення з процедури в основну програму |
| min |

**Стан 2. Виконання процедури KozarezovCalcFirst**

|  |
| --- |
| max |
| ESI — індекс елемента масиву (передача параметра C) EIP — адреса повернення з процедури |
| min |

**Стан 3. ret з процедури KozarezovCalcFirst**

|  |
| --- |
| max |
| Повернення в основну програму, регістри незмінені |
| min |

**Стан 4. push dword ptr KozarezovDataD[esi\*8+4]**

|  |
| --- |
| max |
| D (старші 4 байти) — параметр D старша частина (QWORD) |
| min |

**Стан 5. push dword ptr KozarezovDataD[esi\*8]**

|  |
| --- |
| max |
| D (молодші 4 байти) — параметр D молодша частина (QWORD) D (старші 4 байти) |
| min |

**Стан 6. call KozarezovCalcSecond**

|  |
| --- |
| max |
| EIP — адреса повернення з процедури в основну програму D (молодші 4 байти) D (старші 4 байти) |
| min |

**Стан 7. push ebp (у KozarezovCalcSecond)**

|  |
| --- |
| max |
| EBP — збережений кадр стеку (старе значення EBP) EIP — адреса повернення D (молодші 4 байти) D (старші 4 байти) |
| min |

**Стан 8. mov ebp, esp (у KozarezovCalcSecond)**  
(EBP оновлено, вказує на поточний кадр стеку)

**Стан 9. ret 8 (повернення з KozarezovCalcSecond)**

|  |
| --- |
| max |
| EIP — адреса повернення в основну програму (Параметри D, 8 байт, зі стеку очищено) |
| min |

**Стан 10. Виклик KozarezovCalcDenom**

|  |
| --- |
| max |
| EIP — адреса повернення з процедури KozarezovCalcDenom |
| min |

**Стан 11. Під час обчислень у KozarezovCalcDenom**

|  |
| --- |
| max |
| KozarezovDenom — загальна змінна (PUBLIC, EXTRN) доступна у всіх модулях EIP — адреса повернення |
| min |

**Стан 12. ret з KozarezovCalcDenom**

|  |
| --- |
| max |
| Повернення в основну програму |
| min |

Висновок

У ході виконання лабораторної роботи були опрацьовані методи модульного програмування. Отримано практичні навички створення процедур та передачі параметрів у них. Було розроблено програму на мові асемблера для обчислення значення арифметичного виразу із застосуванням команд співпроцесора. Програма містить три процедури з різними способами передачі параметрів: через регістри, через стек, а також із використанням директив PUBLIC і EXTERN.

Завчасно проведено контрольні розрахунки для шести різних випадків:

1. чисельник і знаменник додатні;
2. чисельник від’ємний, знаменник додатній;
3. чисельник додатній, знаменник від’ємний;
4. чисельник і знаменник від’ємні;
5. порушення області визначення tg(π/2).

Отримані результати програмного виконання незначно відрізняються від контрольних через округлення чисел до шести знаків після коми. В цілому результати збігаються, що підтверджує правильність роботи процедур і всієї програми. Додатково було проаналізовано стан стеку під час передачі параметрів через нього. Виявлено, що параметри доступні через регістр EBP, який встановлюється у пролозі процедури та спочатку дорівнює значенню ESP. При виклику процедури через команду call у стек також додається значення регістру EIP, що містить адресу повернення в основну програму. Команда ret зчитує цю адресу зі стеку, забезпечуючи повернення управління до викликаючої частини коду.