**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 ІІ курсу ФІОТ доц. Павлов В. Г.

групи ІМ-12

Сутулов Нікіта Олегович

номер у списку групи (варіант): 20

номер залікової книжки: 1229

Київ 2023

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

Завдання за варіантом:

номер у списку групи: 20, тож 20 варіант: (arctg(2\*c) / d + 2) / (b - a - 1)

**Контрольні приклади:**

1. Чисельник і знаменник більші за нуль.

a = 4.1, b = 5.4, c = 3.3, d = 1.5.

(arctg(2\*3.3) / 1.5 + 2) / (5.4 - 4.1 - 1) = (arctg(6.6) / 1.5 + 2) / 0.3 =

= (1.42042489879 / 1.5 + 2) / 0.3 = (0.946949932525 + 2) / 0.3 =

= 2.946949932525 / 0.3 = 9.82316644175

Таким чином:

Результат = 9.82316644175

2. Чисельник та знаменник менші за нуль.

a = -5.9, b = -7.4, c = 4.3, d = -0.2.

(arctg(2\*4.3) / (-0.2) + 2) / (-7.4 + 5.9 - 1) = (arctg(8.6) / (-0.2) + 2) / (-2.5) =

= (1.45503710907 / (-0.2) + 2) / (-2.5) = (-7.27518554537 + 2) / (-2.5) =

= -5.27518554537 / (-2.5) = 2.11007421815

Таким чином:

Результат = 2.11007421815

3. Знаменник дорівнює нулю.

a = 19.6, b = 20.6, c = -40.5, d = 19.2.

b - a - 1 = 20.6 - 19.6 - 1 = 1 - 1 = 0.

4. Чисельник більший за нуль, а знаменник менший за нуль.

a = 3.9, b = 4.3, c = -5.4, d = -2.5.

(arctg(2\*(-5.4)) / (-2.5) + 2) / (4.3 - 3.9 - 1) = (arctg(-10.8) / (-2.5) + 2) / (-0.6) =

= (-1.47846699206 / (-2.5) + 2) / (-0.6) = (0.591386796825 + 2) / (-0.6) =

= 2.591386796825 / (-0.6) = -4.31897799471

Таким чином:

Результат = -4.31897799471

5. Чисельник менший за нуль, а знаменник більший за нуль.

a = 2.8, b = 7.9, c = -8.8, d = 0.1.

(arctg(2\*(-8.8)) / 0.1 + 2) / (7.9 - 2.8 - 1) = (arctg(-17.6) / 0.1 + 2) / 4.1 =

= (-1.51403916897 / 0.1 + 2) / 4.1 = (-15.1403916897 + 2) / 4.1 =

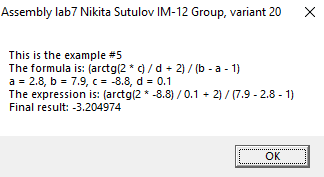
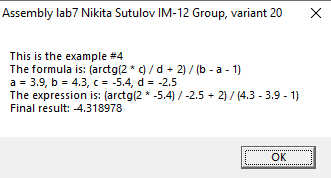
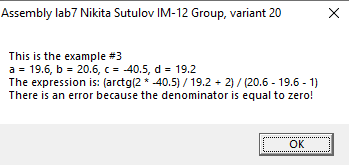
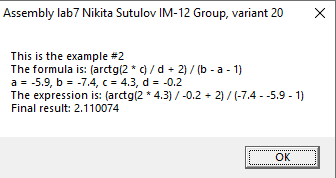
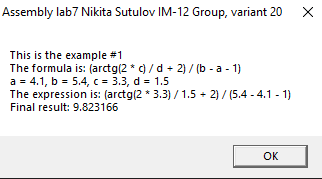
= -13.1403916897 / 4.1 = -3.20497358286

Таким чином:

Результат = -3.20497358286

У формулі використовується функція arctg, яка визначена для всієї множини дійсних чисел, а тому додаткова перевірка відповідності аргумента функції її області визначення, а разом із нею й відповідний контрольний приклад, де цю область визначення порушено, НЕ ПОТРІБНІ.

**Демонстрація роботи програми**



В результаті порівняння контрольних розрахунків з результатами виконання програми видно, що дані збігаються, однак не повністю, а з деякою точністю, так як при контрольних розрахунках виконано округлення до 11 знаків після коми, а під час виконання програми комп’ютер здійснює округлення до 6 знаків після коми.

Отже програма працює коректно в усіх сценаріях:

* чисельник та знаменник більше за 0;
* чисельник та знаменник менше за 0;
* чисельник більше за 0, а знаменник – менше;
* чисельник менше за 0, а знаменник – більше;
* і також той випадок, коли знаменник дорівнює 0, опрацьовується коректно (виводиться повідомлення з помилкою). Тут варто також зауважити, що в цьому випадку не відбувається блокування розрахунків для наступних прикладів (якщо в них знаменник не дорівнює нулю), тож у них результат виконання програми коректний.

**Лістинг програми**

**(операції зі стеком виділено синім кольором, у коментарях позначено, таблиці якого стану стеку відповідає результат виконання цієї операціїї)**

7-20-IM-12-Sutulov.asm:

.386

.model flat, stdcall

option casemap :none

include \masm32\include\masm32rt.inc

public SutulovCurrentB, SutulovCurrentA, SutulovConstOne

extern SutulovGetDenominator:PROTO

.data

; arrays

SutulovAArray dq 4.1, -5.9, 19.6, 3.9, 2.8

SutulovBArray dq 5.4, -7.4, 20.6, 4.3, 7.9

SutulovCArray dq 3.3, 4.3, -40.5, -5.4, -8.8

SutulovDArray dq 1.5, -0.2, 19.2, -2.5, 0.1

; window caption

SutulovWindowCaption db "Assembly lab7 Nikita Sutulov IM-12 Group, variant 20", 0

; window text template

SutulovExampleForm db "This is the example #%d", 0

SutulovFormula db "The formula is: (arctg(2 \* c) / d + 2) / (b - a - 1)", 0

SutulovCurrentValuesForm db "a = %s, b = %s, c = %s, d = %s", 0

SutulovExpressionForm db "The expression is: (arctg(2 \* %s) / %s + 2) / (%s - %s - 1)", 0

SutulovFinalResultForm db "Final result: %s", 0

SutulovErrorMessage db "There is an error because the denominator is equal to zero!", 0

SutulovFinalForm db "%s", 10 ,13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 0

SutulovErrorForm db "%s", 10, 13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 0

SutulovErrorMessageForm db "%s", 0

SutulovConstZero dq 0.0

SutulovConstOne dq 1.0

SutulovConstTwo dq 2.0

SUTULOV\_EIGHT\_BYTES\_OFFSET equ 8

.data?

; buffers

SutulovExampleBuffer db 64 dup (?)

SutulovCurrentValuesBuffer db 64 dup (?)

SutulovExpressionBuffer db 128 dup (?)

SutulovFinalResultBuffer db 64 dup (?)

SutulovFinalBuffer db 256 dup (?)

SutulovDenominator dt ?

SutulovNumerator dt ?

SutulovArctanResult dt ?

SutulovArctanArgument dt ?

SutulovCurrentA dq 1 dup (?)

SutulovCurrentB dq 1 dup (?)

SutulovFinalResult dq ?

SutulovCurrentAString db 16 dup (?)

SutulovCurrentBString db 16 dup (?)

SutulovCurrentCString db 16 dup (?)

SutulovCurrentDString db 16 dup (?)

SutulovFinalResultString db 16 dup (?)

.code

SutulovGetNumeratorFirstTerm proc

fld qword ptr [ecx]

fld qword ptr [eax]

fmul

fstp SutulovArctanArgument

; calculating arctg(2 \* c)

fld SutulovArctanArgument

fld qword ptr [edx]

fpatan

; saving arctan calculation result to 10-bytes buffer

fstp SutulovArctanResult

; calculating numerator: (arctg(2 \* c) / d + 2)

fld SutulovArctanResult

fld qword ptr [ebx]

fdiv ; d can't be 0 according to the task, so no zero checks needed here

ret ; stack state 4

SutulovGetNumeratorFirstTerm endp

SutulovGetNumeratorSecondTerm proc

push ebp ; stack state 7

mov ebp, esp

mov eax, [ebp+SUTULOV\_EIGHT\_BYTES\_OFFSET]

fld qword ptr [eax]

pop ebp ; stack state 8

ret 4 ; stack state 9

SutulovGetNumeratorSecondTerm endp

main:

mov edi, 0 ; counter for indexes

mov esi, 1 ; counter for example numbers

loopToIterateThroughArrays:

cmp edi, 5

je exitLoop

finit

fld SutulovBArray[edi\*SUTULOV\_EIGHT\_BYTES\_OFFSET]

fld SutulovAArray[edi\*SUTULOV\_EIGHT\_BYTES\_OFFSET]

fstp SutulovCurrentA

fstp SutulovCurrentB

call SutulovGetDenominator ; stack state 1

; checking denominator for zero:

fcom SutulovConstZero

fstsw ax

sahf

je denominatorZero

; saving denominator to 10-bytes buffer

fstp SutulovDenominator

lea eax, SutulovCArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET]

lea ebx, SutulovDArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET]

lea ecx, SutulovConstTwo

lea edx, SutulovConstOne

call SutulovGetNumeratorFirstTerm ; stack state 3

push ecx ; stack state 5

call SutulovGetNumeratorSecondTerm ; stack state 6

fadd

; saving numerator to 10-bytes buffer

fstp SutulovNumerator

; dividing numerator by denominator

fld SutulovNumerator

fld SutulovDenominator

fdiv

; saving the final result to 8-bytes variable

fstp SutulovFinalResult

normalWindow:

; converting float numbers to strings for them to be shown correctly

invoke FloatToStr, SutulovAArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET], offset SutulovCurrentAString

invoke FloatToStr, SutulovBArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET], offset SutulovCurrentBString

invoke FloatToStr, SutulovCArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET], offset SutulovCurrentCString

invoke FloatToStr, SutulovDArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET], offset SutulovCurrentDString

invoke FloatToStr, SutulovFinalResult, offset SutulovFinalResultString

; making the window text

invoke wsprintf, offset SutulovExampleBuffer,

offset SutulovExampleForm, esi

invoke wsprintf, offset SutulovCurrentValuesBuffer,

offset SutulovCurrentValuesForm,

offset SutulovCurrentAString, offset SutulovCurrentBString,

offset SutulovCurrentCString, offset SutulovCurrentDString

invoke wsprintf, offset SutulovExpressionBuffer,

offset SutulovExpressionForm,

offset SutulovCurrentCString, offset SutulovCurrentDString,

offset SutulovCurrentBString, offset SutulovCurrentAString

invoke wsprintf, offset SutulovFinalResultBuffer,

offset SutulovFinalResultForm, offset SutulovFinalResultString

invoke wsprintf, offset SutulovFinalBuffer, offset SutulovFinalForm,

offset SutulovExampleBuffer, offset SutulovFormula,

offset SutulovCurrentValuesBuffer, offset SutulovExpressionBuffer,

offset SutulovFinalResultBuffer

invoke MessageBox, 0, offset SutulovFinalBuffer, offset SutulovWindowCaption, 0

inc edi

inc esi

jmp loopToIterateThroughArrays

denominatorZero:

; converting float numbers to strings for them to be shown correctly

invoke FloatToStr, SutulovAArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET], offset SutulovCurrentAString

invoke FloatToStr, SutulovBArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET], offset SutulovCurrentBString

invoke FloatToStr, SutulovCArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET], offset SutulovCurrentCString

invoke FloatToStr, SutulovDArray[edi \* SUTULOV\_EIGHT\_BYTES\_OFFSET], offset SutulovCurrentDString

invoke wsprintf, offset SutulovExampleBuffer,

offset SutulovExampleForm, esi

invoke wsprintf, offset SutulovCurrentValuesBuffer,

offset SutulovCurrentValuesForm,

offset SutulovCurrentAString, offset SutulovCurrentBString,

offset SutulovCurrentCString, offset SutulovCurrentDString

invoke wsprintf, offset SutulovExpressionBuffer,

offset SutulovExpressionForm,

offset SutulovCurrentCString, offset SutulovCurrentDString,

offset SutulovCurrentBString, offset SutulovCurrentAString

invoke wsprintf, offset SutulovFinalResultBuffer,

offset SutulovErrorMessageForm, offset SutulovErrorMessage

invoke wsprintf, offset SutulovFinalBuffer, offset SutulovErrorForm,

offset SutulovExampleBuffer,

offset SutulovCurrentValuesBuffer,

offset SutulovExpressionBuffer,

offset SutulovFinalResultBuffer

invoke MessageBox, 0, offset SutulovFinalBuffer, offset SutulovWindowCaption, 0

inc edi

inc esi

jmp loopToIterateThroughArrays

exitLoop:

invoke ExitProcess, 0

end main

7-20-IM-12-Sutulov-publicextern.asm:

.386

.model flat, stdcall

option casemap :none

public SutulovGetDenominator

extern SutulovCurrentB:qword, SutulovCurrentA:qword, SutulovConstOne:qword

.code

SutulovGetDenominator proc

; calculating denominator: b - a - 1

fld SutulovCurrentB

fld SutulovCurrentA

fsub

fld SutulovConstOne

fsub

ret ; stack state 2

SutulovGetDenominator endp

end

7-20-IM-12-Sutulov.bat:

@echo off

ml /c /coff "7-20-IM-12-Sutulov-publicextern.asm"

ml /c /coff "7-20-IM-12-Sutulov.asm"

link /subsystem:windows "7-20-IM-12-Sutulov.obj" "7-20-IM-12-Sutulov-publicextern.obj"

7-20-IM-12-Sutulov.exe

**Стани стеку під час виконання програми**

| Stack state 1 – call SutulovGetDenominator | |
| --- | --- |
| max |  |
| … | … |
| ***EIP*** | ***Адреса повернення з процедури в основну програму*** |
|  |  |
| min |  |

| Stack state 2 – командою ret у кінці виконання процедури SutulovGetDenominator було витягнуто зі стеку адресу повернення з процедури в основну програму | |
| --- | --- |
| max |  |
| … | … |
|  |  |
|  |  |
| min |  |

| Stack state 3 – call SutulovGetNumeratorFirstTerm | |
| --- | --- |
| max |  |
| … | … |
| ***EIP*** | ***Адреса повернення з процедури в основну програму*** |
|  |  |
| min |  |

| Stack state 4 – командою ret у кінці виконання процедури SutulovGetNumeratorFirstTerm було витягнуто зі стеку адресу повернення з процедури в основну програму | |
| --- | --- |
| max |  |
| … | … |
|  |  |
|  |  |
| min |  |

| Stack state 5 – push ecx | |
| --- | --- |
| max |  |
| … | … |
| ***ECX*** | ***Адреса константи дійсного числа SutulovConstTwo*** |
|  |  |
| min |  |

| Stack state 6 – call SutulovGetNumeratorSecondTerm | |
| --- | --- |
| max |  |
| … | … |
| ***ECX*** | ***Адреса константи дійсного числа SutulovConstTwo*** |
| ***EIP*** | ***Адреса повернення з процедури в основну програму*** |
|  |  |
| min |  |

| Stack state 7 – push ebp | |
| --- | --- |
| max |  |
| … | … |
| ***ECX*** | ***Адреса константи дійсного числа SutulovConstTwo*** |
| ***EIP*** | ***Адреса повернення з процедури в основну програму*** |
| ***EBP*** | ***Поточне значення регістра EBP*** |
|  |  |
| min |  |

| Stack state 8 – pop ebp – наприкінці виконання процедури SutulovGetNumeratorSecondTerm повертаємо зі стеку значення EBP | |
| --- | --- |
| max |  |
| … | … |
| ***ECX*** | ***Адреса константи дійсного числа SutulovConstTwo*** |
| ***EIP*** | ***Адреса повернення з процедури в основну програму*** |
|  |  |
| min |  |

| Stack state 9 – командою ret 4 у кінці виконання процедури SutulovGetNumeratorSecondTerm було витягнено зі стеку адресу повернення з процедури в основну програму, а також зсунуто стек на 4 байти, завдяки чому його було повернено в початковий стан | |
| --- | --- |
| max |  |
| … | … |
|  |  |
| min |  |

**Висновки**

У результаті порівняння контрольних розрахунків із результатами виконання програми для всіх передбачених лабораторною роботою сценаріїв (перевірка аргумента функції на відповідність її області визначення, а також відповідний приклад, у якому цю область визначення порушено, реалізовано не було, оскільки в моєму варіанті використовується функція arctg, яка визначена для всієї множини дійсних чисел, а тому такої перевірки не потребує):

* чисельник та знаменник більше за 0;
* чисельник та знаменник менше за 0;
* чисельник більше за 0, а знаменник – менше;
* чисельник менше за 0, а знаменник – більше;
* знаменник дорівнює 0,

бачимо, що результати обчислень співпадають з деякою точністю (похибка виникає через те, що в ході виконання програми округлення здійснювалось із меншою точністю, ніж у контрольних розрахунках), і випадок, коли знаменник дорівнює 0, обробляється коректно (і в цьому випадку не блокується виконання розрахунків для наступних прикладів, якщо вони відповідають умові, що знаменник не дорівнює нулю).

У ході виконання роботи програму з лабораторної роботи №6 було перероблено так, що процес обчислення виразу було розбито на три процедури, для кожної з яких використовується свій спосіб передачі параметрів (через регістри, через стек, за допомогою директив extern та public). Враховуючи результати порівняння виконання програми та контрольних розрахунків, можна зробити висновок, що всі три процедури функціонують коректно. Також було досліджено стани стеку під час виконання програми й виклику процедур. При передачі параметрів у процедуру через стек, всередині процедури вони доступні за допомогою звернення до регістра ebp (він же base pointer), який встановлюється в початку виконання процедури та значення якого рівне значенню регістра esp (він же stack pointer). Також при виклику кожної процедури в стек додається значення регістра EIP, яке являє собою адресу повернення в основну програму з процедури. При виході з процедури за допомогою команди ret ця адреса повертається зі стеку для подальшого повернення в основну програму.

Таким чином, роблю висновок, що програму написано коректно.

У ході виконання цієї лабораторної роботи мною було вивчено прийоми модульного програмування, а також методи звернення до процедур та передачі в них параметрів (через регістри, стек або за допомогою директив public та extern).