**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-22

Куц Іван Васильович

номер у списку групи (варіант): 12

Київ 2024

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

Завдання за варіантом:

номер у списку групи: 12, формула - .

**Контрольні приклади:**

1. Чисельник і знаменник більші за нуль.

a = 4.1 b = 5.2 c = 3.3 d = 0.5

2. Чисельник та знаменник менші за нуль.

a = -13.9 , b = 1.2, c = 4.3, d = 100.2 .

3. Знаменник дорівнює нулю.

a = 0.5, b = 0.5, c = -40.5, d = 19.2.

.

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

a = 1.3 , b = -12.3, c = 2.1, d = 1.4.

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

a = 1.3 , b = 10.3, c = 2.1, d = 7.45.

6.Від’ємне значення під коренем

a = 1.3 , b = 10.3, c = -2.1, d = 7.45.

У формулі є 2 обмеження – аргумент виразу під знаком кореня має бути область визначення D(x)>= 0, в той час як знаменник функції не може дорівнювати 0

**Демонстрація роботи програми**

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований описЗображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

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 6 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

випадок, коли під коренем знаходиться від’ємне значення. В цьому випадку програма показує повідомлення про помилку та не блокує подальше виконання коду.

**Лістинг програми**

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

7-12-IM-22-Kuts.asm:

.386

.model flat, stdcall

option casemap :none

public AArray, BArray, one

extern calcDenominator:PROTO

include \masm32\include\masm32rt.inc

.data

; arrays

AArray dq 4.1, -13.9, 0.5, 1.3, 1.3, 1.3

BArray dq 5.2, 1.2, 0.5, -12.3, 10.3, 10.3

CArray dq 3.3, 4.3, -40.5, 2.1, 2.1, -2.1

DArray dq 0.5, 100.2, 19.2, 1.4, 7.45, 7.45

; window caption

windowCaption db "System programming, laboratory 7, Ivan Kuts, variant 12", 0

; window text template

testTemplate db "This is the test #%d", 0

formula db "Formula: (sqrt(25/c) - d + 2) / (b + a - 1)", 0

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

numDenomTemplate db "num = %s, denom = %s", 0

expressionTemplate db "The expression is: (sqrt(25/%s) - %s + 2) / (%s + %s - 1)", 0

finalResultsTemplate db "Final result: %s", 0

DenomanitorZeroErrorMessage db "Error: the denominator is equal to zero!", 0

NegativeSqrtErrorMessage db "Error: the value under square root is negative!", 0

finalFormTemplate db "%s", 10 ,13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 0

errorFormTemplate db "%s", 10, 13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 0

errorMessageTemplate db "%s", 0

zero dq 0.0

one dq 1.0

two dq 2.0

twentyFive dq 25.0

.data?

; buffers

numDenomBuffer db 64 dup (?)

testBuffer db 64 dup (?)

currentValuesBuffer db 64 dup (?)

expressionBuffer db 128 dup (?)

finalResultBuffer db 64 dup (?)

finalBuffer db 256 dup (?)

firstTerm dt ?

secondTerm dt ?

numerator dt ?

denominator dt ?

finalResult dq ?

finalResultString db 16 dup (?)

currentAString db 16 dup (?)

currentBString db 16 dup (?)

currentCString db 16 dup (?)

currentDString db 16 dup (?)

.code

calculateFirstTerm proc

fld qword ptr [edx]

fdiv qword ptr [eax + edi \* 8]

fsqrt

ftst

fnstsw ax

sahf

test ah, 01000000b

jnz negativeSqrtValue

ret ; stack state 4

calculateFirstTerm endp

; 2 - d

calculateSecondTerm proc

push ebp ; stack state 8

mov ebp, esp

mov eax, [ebp + 12] ; loading DArray

mov ebx, [ebp + 8]

fld qword ptr [ebx]

fld qword ptr [eax + edi \* 8] ;load current DArray item onto stack

;fstp finalResult

fsub

pop ebp ; stack state 9

ret 8 ; stack state 10

calculateSecondTerm endp

lab7:

mov edi, 0 ; counter for indexes

mov esi, 1 ; counter for example numbers

iterateThroughArrays:

cmp edi, 6

je exitLoop

finit

mov edx, offset twentyFive

call calcDenominator ; stack state 1

; checking denominator for zero:

fcom zero

fstsw ax

sahf

je denominatorZero

;save correct denom

fstp denominator

mov eax, offset CArray

call calculateFirstTerm ; stack state 3

fstp firstTerm

push offset DArray ; stack state 5

push offset two ; stack state 6

call calculateSecondTerm ; stack state 7

fstp secondTerm

fld firstTerm

fld secondTerm

fadd

fstp numerator

fld numerator

fld denominator

fdiv

; saving the final result to 8-bytes variable

fstp finalResult

resultWindow:

invoke FloatToStr, AArray[edi \* 8], offset currentAString

invoke FloatToStr, BArray[edi \* 8], offset currentBString

invoke FloatToStr, CArray[edi \* 8], offset currentCString

invoke FloatToStr, DArray[edi \* 8], offset currentDString

invoke FloatToStr, finalResult, offset finalResultString

; making the window text

invoke wsprintf, offset testBuffer,

offset testTemplate, esi

invoke wsprintf, offset currentValuesBuffer,

offset currentValuesTemplate,

offset currentAString, offset currentBString,

offset currentCString, offset currentDString

invoke wsprintf, offset expressionBuffer,

offset expressionTemplate,

offset currentCString, offset currentDString,

offset currentBString, offset currentAString

invoke wsprintf, offset finalResultBuffer,

offset finalResultsTemplate, offset finalResultString

invoke wsprintf, offset finalBuffer, offset finalFormTemplate,

offset testBuffer, offset formula,

offset currentValuesBuffer, offset expressionBuffer,

offset finalResultBuffer

invoke MessageBox, 0, offset finalBuffer, offset windowCaption, 0

inc edi

inc esi

jmp iterateThroughArrays

negativeSqrtValue:

invoke FloatToStr, AArray[edi \* 8], offset currentAString

invoke FloatToStr, BArray[edi \* 8], offset currentBString

invoke FloatToStr, CArray[edi \* 8], offset currentCString

invoke FloatToStr, DArray[edi \* 8], offset currentDString

invoke wsprintf, offset testBuffer,

offset testTemplate, esi

invoke wsprintf, offset currentValuesBuffer,

offset currentValuesTemplate,

offset currentAString, offset currentBString,

offset currentCString, offset currentDString

invoke wsprintf, offset expressionBuffer,

offset expressionTemplate,

offset currentCString, offset currentDString,

offset currentBString, offset currentAString

invoke wsprintf, offset finalResultBuffer,

offset errorMessageTemplate, offset NegativeSqrtErrorMessage

invoke wsprintf, offset finalBuffer, offset errorFormTemplate,

offset testBuffer,

offset currentValuesBuffer,

offset expressionBuffer,

offset finalResultBuffer

invoke MessageBox, 0, offset finalBuffer, offset windowCaption, 0

inc edi

inc esi

jmp iterateThroughArrays

denominatorZero:

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

invoke FloatToStr, AArray[edi \* 8], offset currentAString

invoke FloatToStr, BArray[edi \* 8], offset currentBString

invoke FloatToStr, CArray[edi \* 8], offset currentCString

invoke FloatToStr, DArray[edi \* 8], offset currentDString

invoke wsprintf, offset testBuffer,

offset testTemplate, esi

invoke wsprintf, offset currentValuesBuffer,

offset currentValuesTemplate,

offset currentAString, offset currentBString,

offset currentCString, offset currentDString

invoke wsprintf, offset expressionBuffer,

offset expressionTemplate,

offset currentCString, offset currentDString,

offset currentBString, offset currentAString

invoke wsprintf, offset finalResultBuffer,

offset errorMessageTemplate, offset DenomanitorZeroErrorMessage

invoke wsprintf, offset finalBuffer, offset errorFormTemplate,

offset testBuffer,

offset currentValuesBuffer,

offset expressionBuffer,

offset finalResultBuffer

invoke MessageBox, 0, offset finalBuffer, offset windowCaption, 0

inc edi

inc esi

jmp iterateThroughArrays

exitLoop:

invoke ExitProcess, 0

end lab7

7-12-IM-22-Kuts.bat

@echo off

ml /c /coff "7-12-IM-22-Kuts-publicextern.asm"

ml /c /coff "7-12-IM-22-Kuts.asm"

link /subsystem:windows "7-12-IM-22-Kuts.obj" "7-12-IM-22-Kuts-publicextern.obj"

7-12-IM-22-Kuts.exe

**Стани стеку під час виконання програми**

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

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

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

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

|  |  |
| --- | --- |
| Stack state 5 – push offset DArray | |
| max |  |
| … | … |
| ***ECX*** | ***Адреса DArray*** |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Stack state 6 – push offset two | |
| max |  |
| … | … |
| ***ECX*** | ***Адреса DArray*** |
| ***ECX*** | ***Адреса two*** |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Stack state 7 – call calculateSecondTerm | |
| max |  |
| … | … |
| ***ECX*** | ***Адреса константи дійсного числа SutulovConstTwo*** |
| ***ECX*** | ***Адреса two*** |
| ***EIP*** | ***Адреса повернення з процедури в основну програму*** |
| … | … |
| min |  |

|  |  |
| --- | --- |
| Stack state 8 – call calculateSecondTerm | |
| max |  |
| … | … |
| ***ECX*** | ***Адреса константи дійсного числа SutulovConstTwo*** |
| ***ECX*** | ***Адреса two*** |
| ***EIP*** | ***Адреса повернення з процедури в основну програму*** |
| ***EBP*** | ***Поточне значення регістра EBP*** |
| … | … |
| min |  |

|  |  |
| --- | --- |
| Stack state 8 – 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 9 – push ebp | |
| max |  |
| … | … |
| ***ECX*** | ***Адреса константи дійсного числа SutulovConstTwo*** |
| ***EIP*** | ***Адреса повернення з процедури в основну програму*** |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Stack state 10 – командою ret 8 у кінці виконання процедури calculateSecondTermбуло 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 теоретичних розрахунків з отриманими результатами можемо сказати, що програма працює коректно. Було протестовано такі випадки:

* 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;
* значення під коренем менше за 0;

бачимо, що результати обчислень співпадають з теоретичними розрахунками, 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).

Під час виконання роботи програму з лабораторної роботи №6 було перероблено так, що процес обчислення виразу розділили на три окремі процедури, кожна з яких використовує свій спосіб передачі параметрів: через регістри, через стек, а також за допомогою директив extern та public. Результати порівняння виконання програми з контрольними розрахунками показали, що всі три процедури функціонують правильно. Крім того, було детально досліджено стани стеку під час виконання програми та виклику процедур. При передачі параметрів через стек, вони стають доступними всередині процедури за допомогою звернення до регістра ebp (base pointer), який встановлюється на початку виконання процедури і дорівнює значенню регістра esp (stack pointer). При виклику кожної процедури в стек додається значення регістра EIP, що являє собою адресу повернення в основну програму з процедури. Під час виходу з процедури за допомогою команди ret ця адреса повертається зі стеку, що дозволяє програмі продовжити виконання з місця виклику процедури. Отже, можна зробити висновок, що програма написана правильно. У процесі виконання цієї лабораторної роботи я засвоїв принципи модульного програмування. Також я навчився різним методам звернення до процедур та передачі в них параметрів, включаючи передачу через регістри, стек і за допомогою директив public та extern.