**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”

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 команд Асемблера для арифметики з плаваючою комою і здобуття навичок виконання розрахунків з елементами масивів.

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

Автоматично згенерований опис

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

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 3 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**

.386

.model flat, stdcall

option casemap :none

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 6, 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 (?)

sqrtResult 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

lab6:

mov edi, 0 ; counter for indexes

mov esi, 1 ; counter for example numbers

iterateThroughArrays:

cmp edi, 6

je exitLoop

finit

; calculating denominator: b + a - 1

fld BArray[edi \* 8]

fld AArray[edi \* 8]

fadd

fld one

fsub

; checking denominator for zero:

fcom zero

fstsw ax

sahf

je denominatorZero

; saving denominator to 10-bytes buffer

fstp denominator

; calculating sqrt(25/c)

fld twentyFive

fld CArray[edi \* 8]

fdiv

fsqrt

ftst

fnstsw ax

sahf

test ah, 01000000b

jnz negativeSqrtValue

; saving sqrt calculation result to 10-bytes buffer

fstp sqrtResult

; calculating numerator: (sqrt(25/c) - d + 2)

fld sqrtResult

fld DArray[edi \* 8]

fsub

fld two

fadd

; saving numerator to 10-bytes buffer

fstp numerator

; dividing numerator by denominator

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 lab6

**Висновки**

У результаті порівняння контрольних розрахунків із результатами виконання програми для всіх передбачених лабораторною роботою сценаріїв:

* 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,;
* від’ємне значення під коренем;

бачимо, що результати обчисленьзбігаються з теоретичними розрахунками з деякою тотчністю, 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). Таким чином, роблю висновок, що програму написано коректно.

Під час виконання цієї лабораторної роботи я ознайомився з командами асемблера MASM32 для роботи з арифметикою з плаваючою комою, такими як додавання (fadd), віднімання (fsub), множення (fmul), ділення (fdiv), порівняння (fcom), обчислення арктангенса (fpatan) тощо. Також я набув навичок обчислення з елементами одновимірних масивів.