**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  
«Технології розроблення системних програм»

на тему  
«Модульне програмування. Використання процедур.»

Виконав: Перевірив:

студент групи ІП-84 доц. кафедри ОТ  
Голубов Іван Олегович Павлов В. Г.  
номер залікової книжки: 8404

Київ 2020

**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.

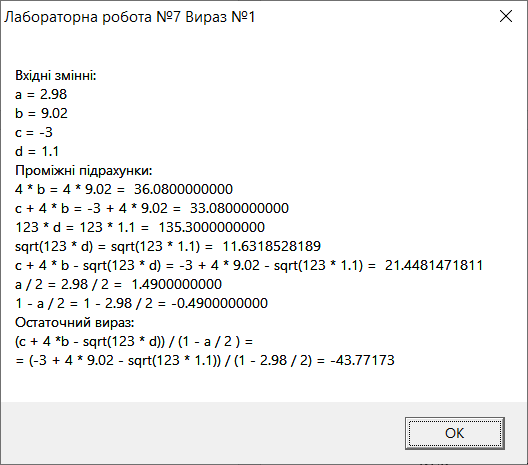
5. Для перевірки правильності виконання розрахунків і результатів, що виводяться, заздалегідь виконати контрольні розрахунки, які повинні охоплювати різноманітні сполучення вхідних даних, на які програма повинна надавати вірну відповідь\*\*\*. Проміжні і остаточні результати контрольних розрахунків привести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

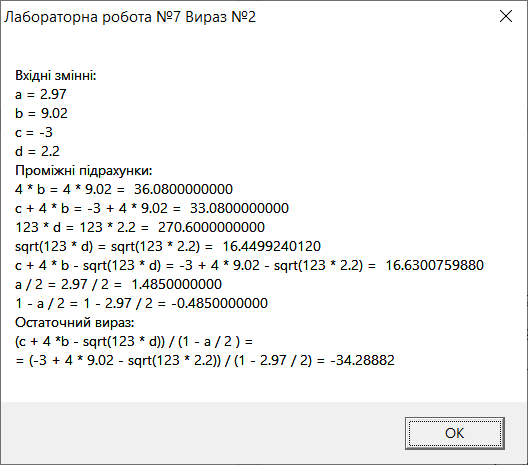
6. Виконати відладку програми шляхом порівняння розрахованих програмою результатів з контрольними прикладами. Лістинг розробленої програми і скріншоти розрахунків по всіх контрольних прикладах привести в звіті по лабораторній роботі.

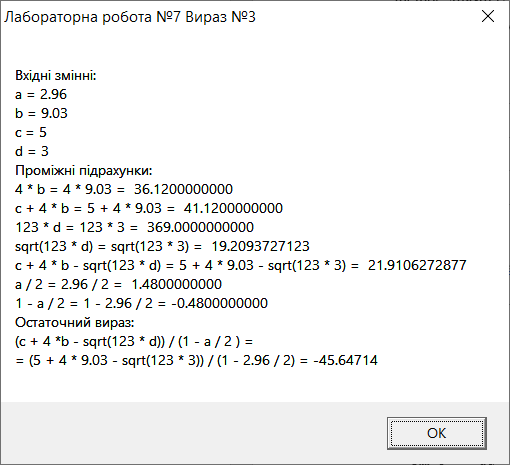
7. У протоколі по лабораторній роботі для першого і другого способів передачі параметрів поруч з відповідними командами у лістингу відобразити в графічному вигляді стани стека при зверненні до процедур, виконання у них команд та повернення з процедур до основної програми.

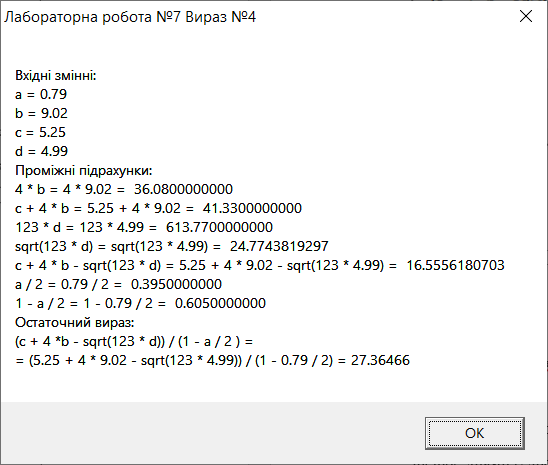
8. Зробити висновки по лабораторній роботі.

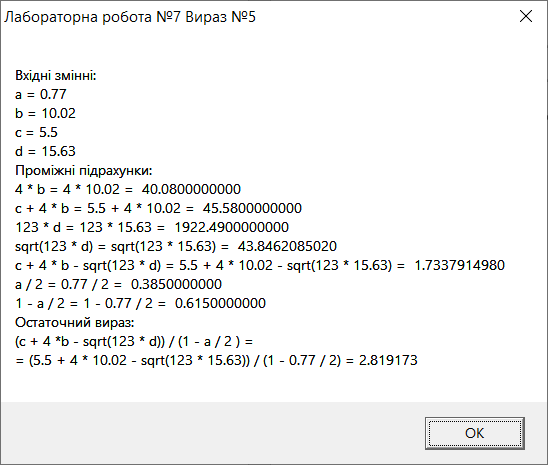
**Скріншоти**



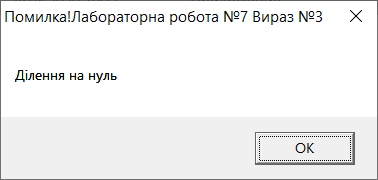




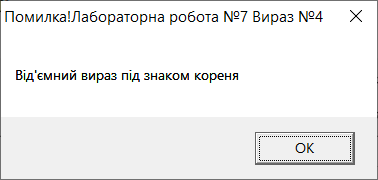




**Скріншот роботи програми при а=2**



**Скріншот роботи програми при d<0**



**Лістинг програми**

**7-03-IP84-Golubov.asm**

.686

.model flat,stdcall

option casemap:none

include \masm32\include\windows.inc ;MB\_OK,NULL

include \masm32\include\kernel32.inc ;ExitProcess

include \masm32\include\user32.inc ;wsprintf, MessageBox

include \masm32\include\masm32.inc ;FloatToStr

include \masm32\include\fpu.inc ;FpuFLtoA

includelib \masm32\lib\kernel32.lib ;ExitProcess

includelib \masm32\lib\user32.lib ;MessageBox

includelib \masm32\lib\masm32.lib ;FloatToStr

includelib \masm32\lib\fpu.lib ;FpuFLtoA

extrn three@0:near

public var\_a, const3, intermedRes6, intermedRes7

.data

arr\_a dq 2.98, 2.97, 2.96, 0.79, 0.77;0,79

arr\_b dq 9.02, 9.02, 9.03, 9.02, 10.02;

arr\_c dq -3.0, -3.0, 5.0, 5.25, 5.5;

arr\_d dq 1.1, 2.2, 3.0, 4.99, 15.63;

var\_a dq ?;

var\_b dq ?;

var\_c dq ?;

var\_d dq ?;

printRes dq ?; В

intermedRes1 DT ?;

intermedRes2 DT ?;

intermedRes3 DT ?;

intermedRes4 DT ?;

intermedRes5 DT ?;

intermedRes6 DT ?;

intermedRes7 DT ?;

RES DT ?;

iter dw 1;

iterator dw -1;

buffa db 25 DUP (?);

buffb db 25 DUP (?);

buffc db 25 DUP (?);

buffd db 25 DUP (?);

buff8 db 250 DUP (?);

buff1 db 250 DUP (?);

buff2 db 250 DUP (?);

buff3 db 250 DUP (?);

buff4 db 250 DUP (?);

buff5 db 250 DUP (?);

buff6 db 250 DUP (?);

buff7 db 250 DUP (?);

buff9 db 250 DUP (?);

buff db 1000 DUP(?), 0;

mainbuff db 1000 DUP(?), 0;

lineFormat db 5 DUP("%s"), 0;

format db "%s",13,"%s",13, "%s",13, "%s",13, "%s",13, "%s",13, "%s",13, "%s",13

mainformat db "%s",13

li DW 50 DUP(?), 0;

lineBuff1 DW 50 DUP(?), 0;

lineBuff2 DW 50 DUP(?), 0;

lineBuff3 DW 50 DUP(?), 0;

lineBuff4 DW 50 DUP(?), 0;

lineBuff5 DW 50 DUP(?), 0;

lineBuff6 DW 50 DUP(?), 0;

lineBuff7 DW 50 DUP(?), 0;

lineBuff8 DW 50 DUP(?), 0;

lineBuff9 DW 100 DUP(?), 0;

count db 50 DUP(?);

.const

const1 dq 4.0;

const2 dq 123.0;

const3 dq 2.0;

const4 dq 0.0;

head db "Лабораторна робота №7 Вираз №%d", 0;

headerror db "Помилка!Лабораторна робота №7 Вираз №%d",0

text1error db "Від'ємний вираз під знаком кореня", 0;

text2error db "Ділення на нуль", 0;

headtask db "Task N", 0;

head1 db "Вхідні змінні:" , 0;

head2 db "Проміжні підрахунки:", 0;

head3 db "Остаточний вираз:", 0;

head4 db "(c + 4 \*b - sqrt(123 \* d)) / (1 - a / 2 ) =", 0;

letter db "%s",13, "a = %s",13,"b = %s", 13, "c = %s", 13,"d = %s", 0;

forma db "%s", 0;

format1 db "%s",13, "4 \* b = 4 \* %s = %s", 0

format2 db "c + 4 \* b = %s + 4 \* %s = %s", 0;

format3 db "123 \* d = 123 \* %s = %s", 0;

format4 db "sqrt(123 \* d) = sqrt(123 \* %s) = %s", 0;

format5 db "c + 4 \* b - sqrt(123 \* d) = %s + 4 \* %s - sqrt(123 \* %s) = %s", 0;

format6 db "a / 2 = %s / 2 = %s", 0;

format7 db "1 - a / 2 = 1 - %s / 2 = %s", 0;

format8 db "%s",13, "%s",13,"= (%s + 4 \* %s - sqrt(123 \* %s)) / (1 - %s / 2) = %s ", 0;

.code

|  |  |
| --- | --- |
| first proc  fld qword ptr[eax]; | eax |
| fld qword ptr[edx]; | edx  eax |
| fmul ; | edx\*eax |
| fstp intermedRes1; проміжні результати ( не впливає на обрахунки, не є параметром обрахунку і результатом) |  |
| fld intermedRes1 | edx\*eax |
| fld qword ptr[ecx] | ecx  edx\*eax |
| fadd ; | ecx+edx\*eax |
| fstp intermedRes2; проміжні результати (не впливає на обрахунки, не є параметром обрахунку і результатом) |  |
| fld intermedRes2 | ecx+edx\*eax |
| ret  first endp |  |
| second proc  ;prolog  push ebp  mov ebp, esp  ;prolog  xor eax, eax  mov eax, [ebp + 12]  xor ecx, ecx  mov ecx, [ebp + 8]  fld qword ptr [eax]; | eax  c+4\*b |
| fld qword ptr [ecx] | ecx  eax  c+4\*b |
| fmul ; | eax\*ecx  c+4\*b |
| fstp intermedRes3; проміжні результати (не впливає на обрахунки, не є параметром обрахунку і результатом) | c+4\*b |
| fld intermedRes3 | eax\*ecx  c+4\*b |
| fsqrt; | sqrt(eax\*ecx)  c+4\*b |
| fstp intermedRes4; проміжні результати (не впливає на обрахунки, не є параметром обрахунку і результатом) | c+4\*b |
| fld intermedRes4 | sqrt(eax\*ecx)  c+4\*b |
| fsub; | c+4\*b-sqrt(eax\*ecx) |
| fstp intermedRes5; проміжні результати ( не впливає на результат обрахунків, не є параметром обрахунку і результатом) |  |
| fld intermedRes5 | c+4\*b - sqrt(123\*d) |
| ;epilog  pop ebp  ;epilog  ret 8  second endp |  |

start:

finit

startLoop:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

xor ebx, ebx;

xor eax, eax

mov bx, iterator;

inc bx;

fld arr\_a[ebx \* 8];

fcom const3

fstsw ax ;сохранение swr в регистре ах

sahf

jz eror1

xor eax, eax

jmp starting1

eror1:

invoke wsprintf, addr count, addr headerror, iter;

invoke MessageBox,NULL, addr text1error, addr count, MB\_OK;

jmp ending

starting1:

fstp var\_a;

fld arr\_b[ebx \* 8];

fstp var\_b;

fld arr\_c[ebx \* 8];

fstp var\_c;

fld arr\_d[ebx \* 8];

fcom const4

fstsw ax ;сохранение swr в регистре ах

sahf

jc eror2

xor eax, eax

jmp starting2

eror2:

invoke wsprintf, addr count, addr headerror, iter;

invoke MessageBox,NULL, addr text2error, addr count, MB\_OK;

jmp ending

starting2:

fstp var\_d;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

xor eax, eax

xor ecx, ecx

xor edx, edx

mov eax, offset const1

mov ecx, offset var\_c

mov edx, offset var\_b

call first;

push offset var\_d

push offset const2

call second;

call three@0

fstp intermedRes6; проміжні результати (не впливає на обрахунки, не є параметром обрахунку і результатом)

fstp intermedRes7; проміжні результати (не впливає на обрахунки, не є параметром обрахунку і результатом)

fld intermedRes7

fdiv; ST(0) <- (c+4\*b- sqrt(123\*d))/(1- a/2)

fstp printRes; DQ printRes <- ST(0)

invoke wsprintf, addr count, addr head, iter;

invoke FloatToStr, var\_c, addr buffc;

invoke FloatToStr, var\_b, addr buffb;

invoke FloatToStr, var\_d, addr buffd;

invoke FloatToStr, var\_a, addr buffa;

invoke FloatToStr, printRes, addr buff8;

invoke FpuFLtoA,offset intermedRes1, 10,offset buff1, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes2, 10,offset buff2, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes3, 10,offset buff3, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes4, 10,offset buff4, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes5, 10,offset buff5, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes6, 10,offset buff6, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes7, 10,offset buff7, SRC1\_REAL

invoke wsprintf, addr lineBuff1,

addr letter,

addr head1,

addr buffa,

addr buffb,

addr buffc,

addr buffd

invoke wsprintf, addr lineBuff2,

addr format1,

addr head2,

addr buffb,

addr buff1

invoke wsprintf, addr lineBuff3,

addr format2,

addr buffc,

addr buffb,

addr buff2;

invoke wsprintf, addr lineBuff4,

addr format3,

addr buffd,

addr buff3;

invoke wsprintf, addr lineBuff5,

addr format4,

addr buffd,

addr buff4;

invoke wsprintf, addr lineBuff6,

addr format5,

addr buffc,

addr buffb,

addr buffd,

addr buff5;

invoke wsprintf, addr lineBuff7,

addr format6,

addr buffa,

addr buff6;

invoke wsprintf, addr lineBuff8,

addr format7,

addr buffa,

addr buff7;

invoke wsprintf, addr lineBuff9,

addr format8,

addr head3,

addr head4,

addr buffc,

addr buffb,

addr buffd,

addr buffa,

addr buff8

invoke wsprintf, addr buff,

addr format,

addr lineBuff1,

addr lineBuff2,

addr lineBuff3,

addr lineBuff4,

addr lineBuff5,

addr lineBuff6,

addr lineBuff7,

addr lineBuff8,

addr lineBuff9;

invoke MessageBox,NULL, addr buff, addr count, MB\_OK;

jmp ending

ending:

inc iter;

inc iterator;

cmp iterator, 4

jnz startLoop

invoke ExitProcess, NULL;

end start

**7-03-IP84-Golubov-Proc.asm**

.686

.model flat, stdcall

option casemap: none

extern var\_a:QWORD, const3:QWORD, intermedRes6:QWORD , intermedRes7:QWORD

public three

.data

.code

three proc

fld1; ST(0) <- c+4\*b - sqrt(123\*d), ST(1) <- 1

fld var\_a; ST(0) <- c+4\*b - sqrt(123\*d), ST(1) <- 1, ST(2) <- a

fld const3; ST(0) <- c+4\*b - sqrt(123\*d), ST(1) <- 1, ST(2) <- a, ST(3) <-2

fdiv; ST(0) <- c+4\*b - sqrt(123\*d), ST(1) <- 1, ST(2) <- a/2

fstp intermedRes6; проміжні результати ( не впливає на результат обрахунків, не є параметром обрахунку і результатом)

fld intermedRes6

fsub; ST(0) <- c+4\*b - sqrt(123\*d), ST(1) <- 1-a/2

fld intermedRes6

ret

three endp

end

**Перевірка результатів розрахунків**

Похибки обчислень виділені **жирним шрифтом**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | №1 | №2 | №3 |
| a=2.98  b=9.02  c=-3  d=1.1 | a=2.97  b=9.02  c=-3  d=2.2 | a=2.96  b=9.03  c=5  d=3 |
| 4\**b* | Власні розрахунки | 36.08 | 36.08 | 36.12 |
| Програма | 36.08 | 36.08 | 36.12 |
| *c*+4\**b* | Власні розрахунки | 33.08 | 33.08 | 41.12 |
| Програма | 33.08 | 33.08 | 41.12 |
|  | Власні розрахунки | 135.3 | 270.6 | 369 |
| Програма | 135.3 | 270.6 | 369 |
|  | Власні розрахунки | 11.6318528**1** | 16.4499240**1** | 19.2093727**1** |
| Програма | 11.6318528 | 16.4499240 | 19.2093727 |
|  | Власні розрахунки | 21.448147**18** | 16.63007**598** | 21.910627**28** |
| Програма | 21.448147**2** | 16.63007**60** | 21.910627**3** |
|  | Власні розрахунки | 1.49 | 1.485 | 1.48 |
| Програма | 1.49 | 1.485 | 1.48 |
|  | Власні розрахунки | -0.49 | -0.485 | -0.48 |
| Програма | -0.49 | -0.485 | -0.48 |
|  | Власні розрахунки | -43.7717**28** | -34.2888**16** | -45.64714**01** |
| Програма | -43.7717**3** | -34.2888**2** | -45.64714 |
| Коректність роботи програми |  | ˅ | ˅ | ˅ |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | №4 | №5 |
| a=0.79  b=9.02  c=5.25  d=4.99 | a=0.77  b=10.02  c=5.5  d=15.63 |
| 4\**b* | Власні розрахунки | 36.08 | 40.08 |
| Програма | 36.08 | 40.08 |
| *c*+4\**b* | Власні розрахунки | 41.33 | 45.58 |
| Програма | 41.33 | 45.58 |
|  | Власні розрахунки | 613.77 | 1922.49 |
| Програма | 613.77 | 1922.49 |
|  | Власні розрахунки | 24.7743819**2** | 43.8462085**02** |
| Програма | 24.7743819 | 43.8462085 |
|  | Власні розрахунки | 16.555618**07** | 1.733791**49** |
| Програма | 16.555618**1** | 1.733791**5** |
|  | Власні розрахунки | 0.395 | 0.385 |
| Програма | 0.395 | 0.385 |
|  | Власні розрахунки | 0.605 | 0.615 |
| Програма | 0.605 | 0.615 |
|  | Власні розрахунки | 27.3646**57** | 2.819173**1** |
| Програма | 27.3646**6** | 2.819173 |
| Коректність роботи програми |  | ˅ | ˅ |

**Висновки**

У ході виконання лабораторної роботи я вивчив прийоми модульного програмування, методи звернення до процедур і передачі в них параметрів.

Я створив програму на мові асемблер, в якій використовувати три процедури з різними способами передачі параметрів: через регістри, стек та за допомогою директив EXTRN та PUBLIC.

Чисельник дробу зі свого варіанту індивідуального завдання до лабораторної роботи 6 розділив на два доданка, з яких для першого застосувати передачу параметрів і результату через регістри, а для другого – через стек. Для знаменника використовав метод оголошення загальних змінних директивами public і extern. Виведення результату виконав в основній програмі. Розрахунки повторив в програмі для 5 значень змінних, причому всі вхідні значення задав дійсними числами у вигляді одновимірних масивів.

Перевірив правильність виконання розрахунків і результатів, що виводяться, заздалегідь виконав контрольні розрахунки. Виконав відладку програми шляхом порівняння розрахованих програмою результатів з контрольними прикладами. Проміжні і остаточні результати контрольних розрахунків привів в таблиці. Програма з заданою точністю змогла чітко розрахувати вираз і зобразити його на екрані з підстановкою значень і результату.

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