**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 №8**

з дисципліни  
«Технології розроблення системних програм»

на тему  
«Розробка і використання динамічних бібліотек.»

Виконав: Перевірив:

студент групи ІП-84 доц. кафедри ОТ  
Сапбиєв Максим Назарович Павлов В. Г.  
номер залікової книжки: 8419

Київ 2020

**Мета роботи**

Вивчення прийомів розробки і використання процедур, представлених у вигляді динамічних бібліотек.

**Порядок виконання роботи**

1. Вивчити методи розробки і використання DLL- бібліотек.

Для свого варіанту індивідуального завдання до лабораторної роботи 6 розробити процедуру на мові Асемблер, в якій реалізувати обчислення заданого арифметичного виразу і виведення результатів у віконному інтерфейсі. Параметрами, які передаються до процедурі, є значення змінних. Процедуру розробити в двох варіантах:

• без вказівки точки входу;

• з використанням точки входу.

3. Розробити файл установок модуля DEF і виконати компіляцію обох варіантів файлу бібліотек DLL. Лістинги обох варіантів написання процедур, файлу DEF і командних рядків компіляції представити в звіті по лабораторній роботі.

4. Написати і скомпілювати файл основної програми, в якому передбачити неявне завантаження розробленої DLL-бібліотеки з передачею в неї в циклі 5 значень змінних, заданих дійсними числами у вигляді одновимірних масивів.

5. Перевірити роботу основної програми з підключенням по черзі обох варіантів DLL- бібліотек і привести в звіті по лабораторній роботі лістинг основної програми і скріншоти вікон з результатами обчислень.

6. Написати і скомпілювати файл основної програми, в якому передбачити явне завантаження розробленої DLL- бібліотеки з передачею до неї в циклі тих же 5 значень змінних.

7. Переконатися в працездатності основної програми при зверненні в ній по черзі до кожної з DLL- бібліотек, розроблених в п. 3. Лістинг основної програми привести в звіті по лабораторній роботі.

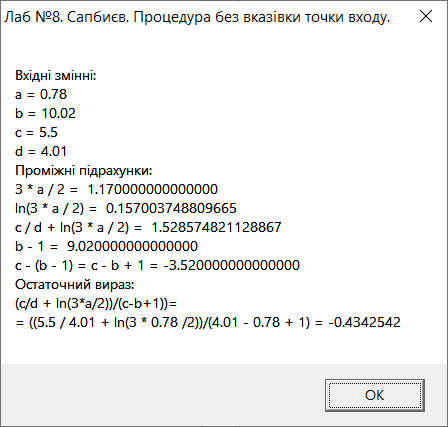
8. Для перевірки правильності виконання розрахунків і результатів, що виводяться, заздалегідь виконати контрольні розрахунки. Проміжні і остаточні результати контрольних розрахунків привести в звіті по лабораторній роботі.

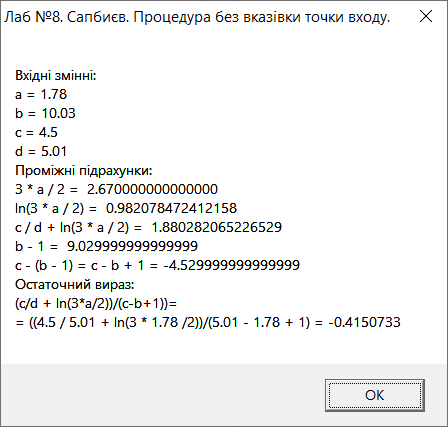
9. Виконати відладку програми шляхом порівняння розрахованих програмою результатів з контрольними прикладами.

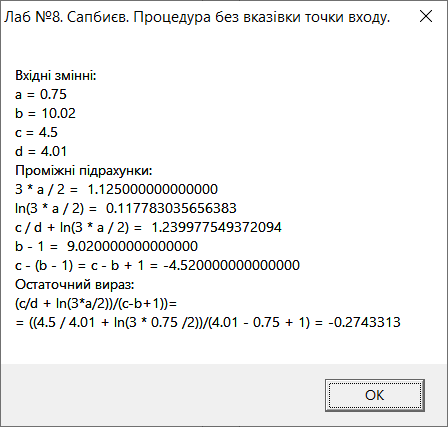
10. Зробити висновки по лабораторній роботі.

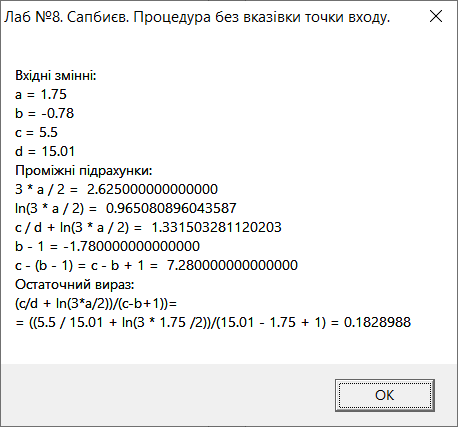
**Скріншоти**

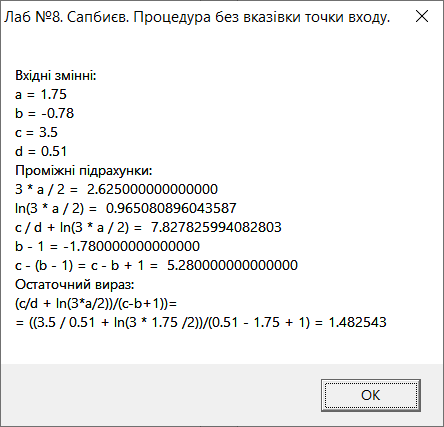
**8-16-IP84-Sapbiev-dynamic-without-point.exe**



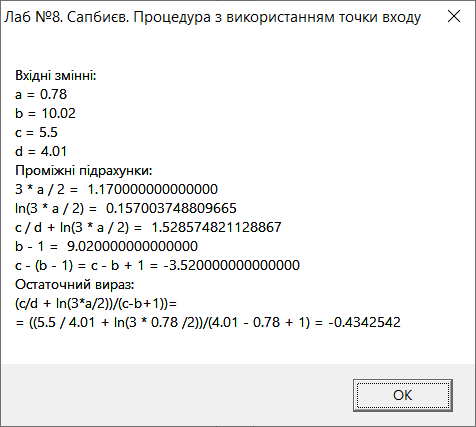


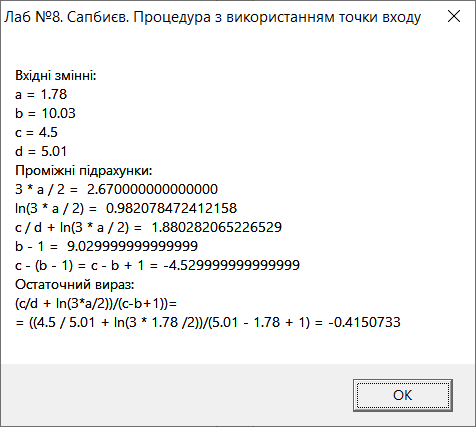


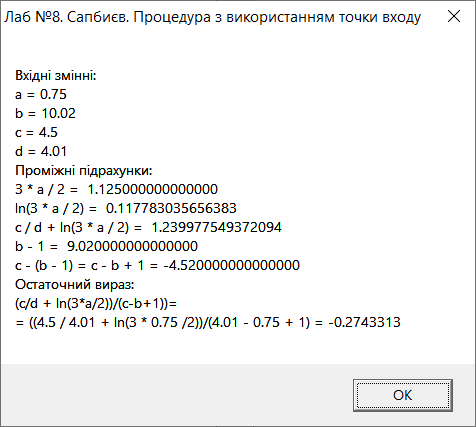


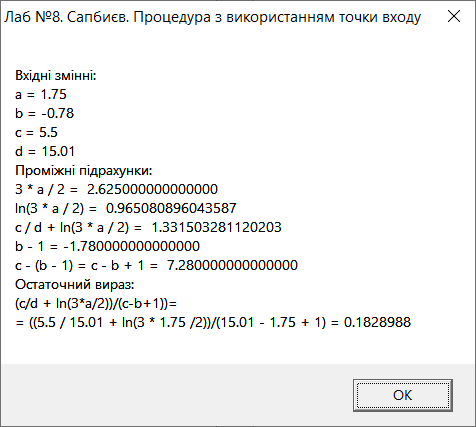


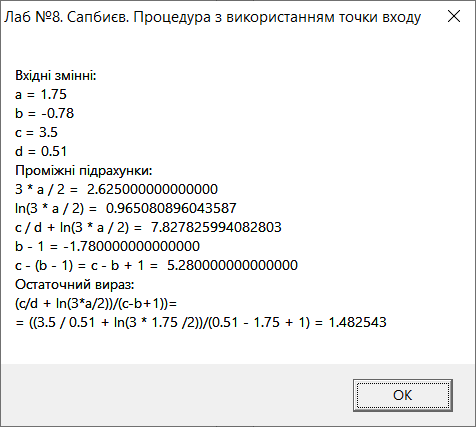
**8-16-IP84-Sapbiev-dynamic-with-point.exe**



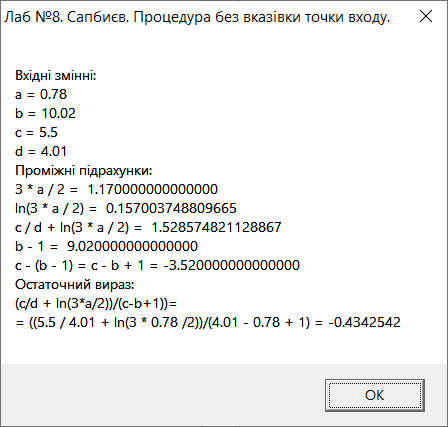


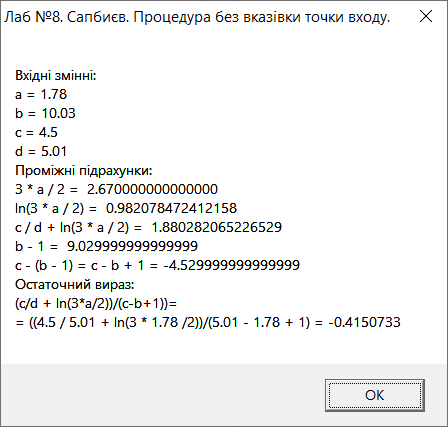


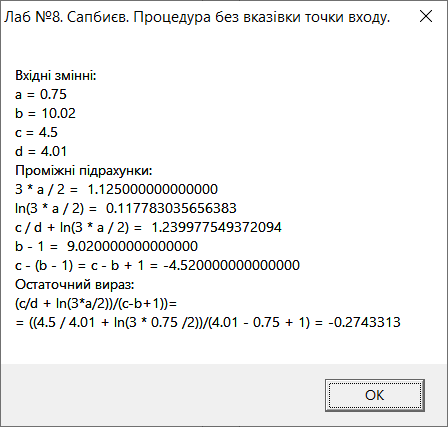


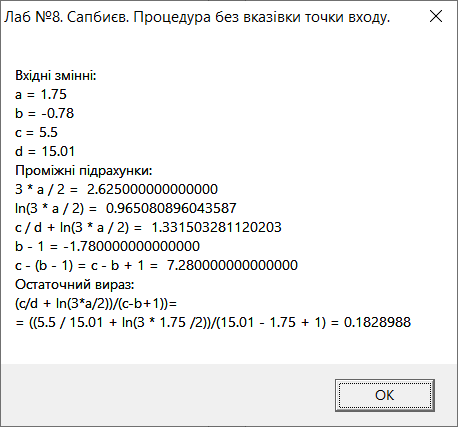


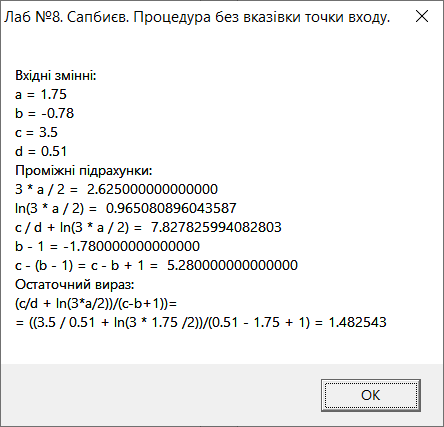
**8-16-IP84-** **Sapbiev-static-without-point.exe**



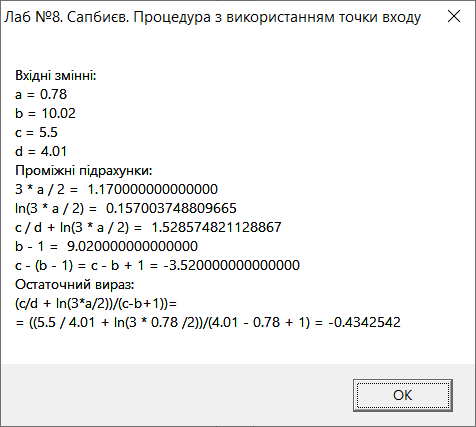


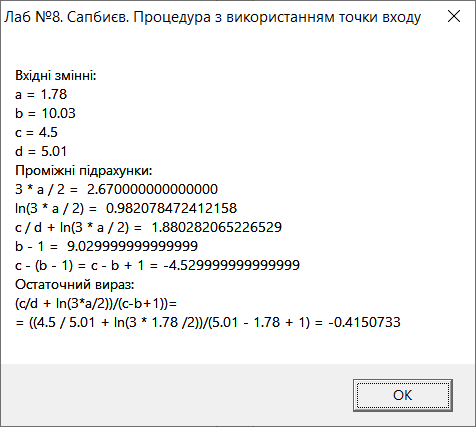


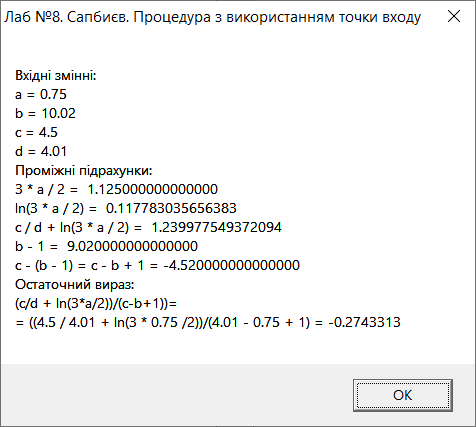


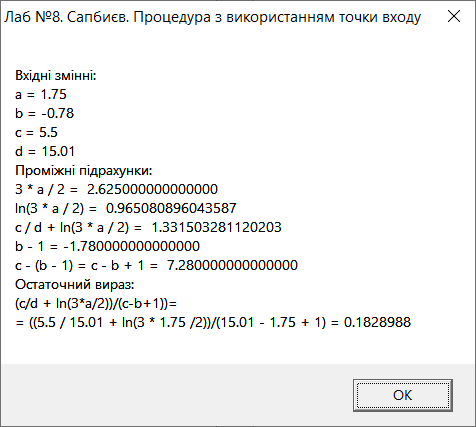


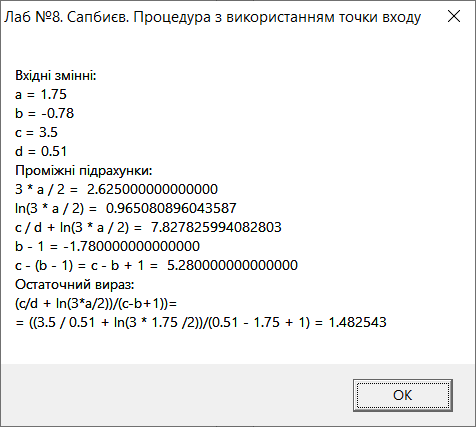
**8-16-IP84-** **Sapbiev-static-with-point.exe**











**Лістинг програми**

**8-16-IP84-Sapbiev-with.def**

LIBRARY 8-16-IP84-Sapbiev-with

EXPORTS WithPoint

**8-16-IP84-Sapbiev-with.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

.data

var\_a dq ?;

var\_b dq ?;

var\_c dq ?;

var\_d dq ?;

printRes dq ?;

intermedRes1 DT ?;

intermedRes2 DT ?;

intermedRes3 DT ?;

intermedRes4 DT ?;

intermedRes5 DT ?;

intermedRes6 DT ?;

intermedRes7 DT ?;

RES DT ?;

temp DT ?;

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 (?);

fi dt ?;

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

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;

count db 150 DUP(?);

.const

const1 dq 3.0;

const2 dq 2.0;

const3 dq 0.0;

head db "Лаб №8. Сапбиєв. Процедура з використанням точки входу", 0;

headerror db "Помилка!Лаб №8. Сапбиєв.",0

text1error db "Від'ємний вираз під логарифмом", 0;

text2error db "Ділення на нуль", 0;

headtask db "TN", 0;

head1 db "Вхідні змінні:" , 0;

head2 db "Проміжні підрахунки:", 0;

head3 db "Остаточний вираз:", 0;

head4 db "(c/d + ln(3\*a/2))/(c-b+1))=", 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, "3 \* a / 2 = %s", 0

format2 db "ln(3 \* a / 2) = %s", 0;

format3 db "c / d + ln(3 \* a / 2) = %s", 0;

format4 db "b - 1 = %s", 0;

format5 db "c - (b - 1) = c - b + 1 = %s", 0;

format6 db "%s",13, "%s",13,"= ((%s / %s + ln(3 \* %s /2))/(%s - %s + 1) = %s ", 0;

.code

DLLEntry proc hInstDLL: HINSTANCE,

reason: DWORD,

reserved: DWORD

mov eax,TRUE

ret

DLLEntry endp

WithPoint proc

push ebp

mov ebp, esp

;prolog

finit

xor ecx, ecx

xor edx, edx

mov ecx, [ebp + 20]

fld qword ptr [ecx];

fstp var\_a;

xor ecx, ecx

mov ecx, [ebp + 12]

mov edx, [ebp + 8]

fld qword ptr [edx];

fstp var\_d;

fld qword ptr [ecx];

fstp var\_c;

xor ecx, ecx

mov ecx, [ebp + 16]

fld qword ptr [ecx];

fstp var\_b;

fld var\_a

fcom const3

fstsw ax

sahf

jc eror1

xor eax, eax

jmp starting1

eror1:

invoke wsprintf, addr count, addr headerror

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

jmp ending

starting1:

fstp var\_a;

fld var\_c

fld1;

fadd;

fcom var\_b

fstsw ax

sahf

jz eror2

xor eax, eax

jmp starting2

eror2:

invoke wsprintf, addr count, addr headerror

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

jmp ending

fstp fi;

fstp fi;

fstp fi;

fstp fi;

starting2:

fld var\_d

fcom const3

fstsw ax

sahf

jz eror3

xor eax, eax

jmp starting3

eror3:

invoke wsprintf, addr count, addr headerror

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

jmp ending

starting3:

fstp var\_d;

fstp fi

fld var\_c

fld var\_d

fdiv;

fld var\_a

fmul const1

fdiv const2

fstp intermedRes1

fld intermedRes1

fstp temp;

fldln2

fld temp;

fyl2x

fstp intermedRes2

fld intermedRes2

fadd

fstp intermedRes3

fld intermedRes3

fld var\_c

fld var\_b

fld1

fsub

fstp intermedRes4

fld intermedRes4

fsub

fstp intermedRes5

fld intermedRes5

fdiv

fstp printRes; DQ printRes <- ST(0)

window proc

invoke wsprintf, addr count, addr head

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, 15,offset buff1, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes2, 15,offset buff2, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes3, 15,offset buff3, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes4, 15,offset buff4, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes5, 15,offset buff5, 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 buff1

invoke wsprintf, addr lineBuff3,

addr format2,

addr buff2

invoke wsprintf, addr lineBuff4,

addr format3,

addr buff3

invoke wsprintf, addr lineBuff5,

addr format4,

addr buff4

invoke wsprintf, addr lineBuff6,

addr format5,

addr buff5,

addr buffb,

addr buffd,

addr buff5;

invoke wsprintf, addr lineBuff7,

addr format6,

addr head3,

addr head4,

addr buffc,

addr buffd,

addr buffa,

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

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

pop ebp

ret 0

window endp

call window

jmp ending

ending:

WithPoint endp

end DLLEntry

**8-16-IP84-Sapbiev.bat**

ml /c /coff "8-16-IP84-Sapbiev-with.asm"

link /DLL /subsystem:windows /DEF:"8-16-IP84-Sapbiev-with.def" "8-16-IP84-Sapbiev-with.obj"

ml /c /coff "8-16-IP84-Sapbiev-without.asm"

link /DLL /DEF:"8-16-IP84-Sapbiev-without.def" /NOENTRY "8-16-IP84-Sapbiev-without.obj"

ml /c /coff "8-16-IP84-Sapbiev-static-with-point.asm"

link /subsystem:windows "8-16-IP84-Sapbiev-static-with-point.obj"

ml /c /coff "8-16-IP84-Sapbiev-static-without-point.asm"

link /subsystem:windows "8-16-IP84-Sapbiev-static-without-point.obj"

ml /c /coff "8-16-IP84-Sapbiev-dynamic-with-point.asm"

link /subsystem:windows "8-16-IP84-Sapbiev-dynamic-with-point.obj"

ml /c /coff "8-16-IP84-Sapbiev-dynamic-without-point.asm"

link /subsystem:windows "8-16-IP84-Sapbiev-dynamic-without-point.obj"

**8-16-IP84-Sapbiev-without.def**

LIBRARY 8-16-IP84-Sapbiev-without

EXPORTS WithoutPoint

**8-16-IP84-Sapbiev-without.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

.data

var\_a dq ?;

var\_b dq ?;

var\_c dq ?;

var\_d dq ?;

printRes dq ?;

intermedRes1 DT ?;

intermedRes2 DT ?;

intermedRes3 DT ?;

intermedRes4 DT ?;

intermedRes5 DT ?;

intermedRes6 DT ?;

intermedRes7 DT ?;

RES DT ?;

temp DT ?;

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 (?);

fi dt ?;

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

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;

count db 150 DUP(?);

.const

const1 dq 3.0;

const2 dq 2.0;

const3 dq 0.0;

head db "Лаб №8. Сапбиєв. Процедура без вказівки точки входу.", 0;

headerror db "Помилка!Лаб №8. Сапбиєв.",0

text1error db "Від'ємний вираз під логарифмом", 0;

text2error db "Ділення на нуль", 0;

headtask db "TN", 0;

head1 db "Вхідні змінні:" , 0;

head2 db "Проміжні підрахунки:", 0;

head3 db "Остаточний вираз:", 0;

head4 db "(c/d + ln(3\*a/2))/(c-b+1))=", 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, "3 \* a / 2 = %s", 0

format2 db "ln(3 \* a / 2) = %s", 0;

format3 db "c / d + ln(3 \* a / 2) = %s", 0;

format4 db "b - 1 = %s", 0;

format5 db "c - (b - 1) = c - b + 1 = %s", 0;

format6 db "%s",13, "%s",13,"= ((%s / %s + ln(3 \* %s /2))/(%s - %s + 1) = %s ", 0;

.code

WithoutPoint proc

push ebp

mov ebp, esp

;prolog

finit

xor ecx, ecx

xor edx, edx

mov ecx, [ebp + 20]

fld qword ptr [ecx];

fstp var\_a;

xor ecx, ecx

mov ecx, [ebp + 12]

mov edx, [ebp + 8]

fld qword ptr [edx];

fstp var\_d;

fld qword ptr [ecx];

fstp var\_c;

xor ecx, ecx

mov ecx, [ebp + 16]

fld qword ptr [ecx];

fstp var\_b;

fld var\_a

fcom const3

fstsw ax

sahf

jc eror1

xor eax, eax

jmp starting1

eror1:

invoke wsprintf, addr count, addr headerror

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

jmp ending

starting1:

fstp var\_a;

fld var\_c

fld1;

fadd;

fcom var\_b

fstsw ax

sahf

jz eror2

xor eax, eax

jmp starting2

eror2:

invoke wsprintf, addr count, addr headerror

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

jmp ending

fstp fi;

fstp fi;

fstp fi;

fstp fi;

starting2:

fld var\_d

fcom const3

fstsw ax

sahf

jz eror3

xor eax, eax

jmp starting3

eror3:

invoke wsprintf, addr count, addr headerror

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

jmp ending

starting3:

fstp var\_d;

fstp fi

fld var\_c

fld var\_d

fdiv;

fld var\_a

fmul const1

fdiv const2

fstp intermedRes1

fld intermedRes1

fstp temp;

fldln2

fld temp;

fyl2x

fstp intermedRes2

fld intermedRes2

fadd

fstp intermedRes3

fld intermedRes3

fld var\_c

fld var\_b

fld1

fsub

fstp intermedRes4

fld intermedRes4

fsub

fstp intermedRes5

fld intermedRes5

fdiv

fstp printRes; DQ printRes <- ST(0)

window proc

invoke wsprintf, addr count, addr head

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, 15,offset buff1, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes2, 15,offset buff2, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes3, 15,offset buff3, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes4, 15,offset buff4, SRC1\_REAL

invoke FpuFLtoA,offset intermedRes5, 15,offset buff5, 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 buff1

invoke wsprintf, addr lineBuff3,

addr format2,

addr buff2

invoke wsprintf, addr lineBuff4,

addr format3,

addr buff3

invoke wsprintf, addr lineBuff5,

addr format4,

addr buff4

invoke wsprintf, addr lineBuff6,

addr format5,

addr buff5,

addr buffb,

addr buffd,

addr buff5;

invoke wsprintf, addr lineBuff7,

addr format6,

addr head3,

addr head4,

addr buffc,

addr buffd,

addr buffa,

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

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

pop ebp

ret 0

window endp

call window

jmp ending

ending:

WithoutPoint endp

end

**8-16-IP84-Sapbiev-dynamic-without-point.asm**

.686

.model flat,stdcall

option casemap:none

include \masm32\include\masm32rt.inc

.const

libNameWithoutPoint CHAR "8-16-IP84-Sapbiev-without", 0h

funcNameWithoutPoint CHAR "WithoutPoint", 0h

.data

arr\_a dq 0.78, 1.78, 0.75, 1.75, 1.75;

arr\_b dq 10.02, 10.03, 10.02, -0.78, -0.78;

arr\_c dq 5.5, 4.5, 4.5, 5.5, 3.5;

arr\_d dq 4.01, 5.01, 4.01, 15.01, 0.51;

count db 50 DUP(?);

var\_a dq ?;

var\_b dq ?;

var\_c dq ?;

var\_d dq ?;

buffc db 25 DUP (?);

head1 db "Бібліотека не знайдена!" , 0;

iter dw 1;

iterator dw -1;

head db "Лаб №8", 0;

.data?

hLibWithoutPoint DWORD ?

lpFuncWithoutPoint DWORD ?

.code

start:

xor eax, eax

invoke LoadLibrary, ADDR libNameWithoutPoint

cmp eax,0

je @error

mov hLibWithoutPoint,eax

invoke GetProcAddress, hLibWithoutPoint, ADDR funcNameWithoutPoint

mov lpFuncWithoutPoint,eax

finit

startLoop:

xor ebx, ebx;

mov bx, iterator;

inc bx;

fld arr\_d[ebx \* 8];

fstp var\_d;

fld arr\_c[ebx \* 8];

fstp var\_c;

fld var\_c

fld arr\_b[ebx \* 8];

fstp var\_b;

fld arr\_a[ebx \* 8];

fstp var\_a;

push offset var\_a

push offset var\_b

push offset var\_c

push offset var\_d

call [lpFuncWithoutPoint]

inc iter;

inc iterator;

cmp iterator, 4

jnz startLoop

jmp @exit

@error:

invoke MessageBox,NULL, addr head1, addr head, MB\_OK;

@exit:

invoke ExitProcess, NULL;

end start

**8-16-IP84-Sapbiev-dynamic-with-point.asm**

.686

.model flat,stdcall

option casemap:none

include \masm32\include\masm32rt.inc

.const

libNameWithPoint CHAR "8-16-IP84-Sapbiev-with", 0h

funcNameWithPoint CHAR "WithPoint", 0h

.data

arr\_a dq 0.78, 1.78, 0.75, 1.75, 1.75;

arr\_b dq 10.02, 10.03, 10.02, -0.78, -0.78;

arr\_c dq 5.5, 4.5, 4.5, 5.5, 3.5;

arr\_d dq 4.01, 5.01, 4.01, 15.01, 0.51;

count db 50 DUP(?);

var\_a dq ?;

var\_b dq ?;

var\_c dq ?;

var\_d dq ?;

buffc db 25 DUP (?);

head1 db "Бібліотека не знайдена!" , 0;

iter dw 1;

iterator dw -1;

head db "Лаб №8", 0;

.data?

hLibWithPoint DWORD ?

lpFuncWithPoint DWORD ?

.code

start:

xor eax, eax

invoke LoadLibrary, ADDR libNameWithPoint

cmp eax,0

je @error

mov hLibWithPoint,eax

invoke GetProcAddress, hLibWithPoint, ADDR funcNameWithPoint

mov lpFuncWithPoint,eax

finit

startLoop:

xor ebx, ebx;

mov bx, iterator;

inc bx;

fld arr\_d[ebx \* 8];

fstp var\_d;

fld arr\_c[ebx \* 8];

fstp var\_c;

fld var\_c

fld arr\_b[ebx \* 8];

fstp var\_b;

fld arr\_a[ebx \* 8];

fstp var\_a;

push offset var\_a

push offset var\_b

push offset var\_c

push offset var\_d

call [lpFuncWithPoint]

inc iter;

inc iterator;

cmp iterator, 4

jnz startLoop

jmp @exit

@error:

invoke MessageBox,NULL, addr head1, addr head, MB\_OK;

@exit:

invoke ExitProcess, NULL;

end start

**8-16-IP84-Sapbiev-static-without-point.asm**

. .686

.model flat,stdcall

option casemap:none

include \masm32\include\masm32rt.inc

includelib \masm32\labs\labs8\8-16-IP84-Sapbiev-without.lib

WithoutPoint Proto

.data

arr\_a dq 0.78, 1.78, 0.75, 1.75, 1.75;

arr\_b dq 10.02, 10.03, 10.02, -0.78, -0.78;

arr\_c dq 5.5, 4.5, 4.5, 5.5, 3.5;

arr\_d dq 4.01, 5.01, 4.01, 15.01, 0.51;

var\_a dq ?;

var\_b dq ?;

var\_c dq ?;

var\_d dq ?;

iter dw 1;

iterator dw -1;

.code

start:

finit

startLoop:

xor ebx, ebx;

mov bx, iterator;

inc bx;

fld arr\_a[ebx \* 8];

fstp var\_a;

fld arr\_b[ebx \* 8];

fstp var\_b;

fld arr\_c[ebx \* 8];

fstp var\_c;

fld arr\_d[ebx \* 8];

fstp var\_d;

push offset var\_a

push offset var\_b

push offset var\_c

push offset var\_d

call WithoutPoint

inc iter;

inc iterator;

cmp iterator, 4

jnz startLoop

invoke ExitProcess, NULL;

end start

**8-16-IP84-Sapbiev-static-with-point.asm**

.686

.model flat,stdcall

option casemap:none

include \masm32\include\masm32rt.inc

includelib \masm32\labs\labs8\8-16-IP84-Sapbiev-with.lib

WithPoint Proto

.data

arr\_a dq 0.78, 1.78, 0.75, 1.75, 1.75;

arr\_b dq 10.02, 10.03, 10.02, -0.78, -0.78;

arr\_c dq 5.5, 4.5, 4.5, 5.5, 3.5;

arr\_d dq 4.01, 5.01, 4.01, 15.01, 0.51;

var\_a dq ?;

var\_b dq ?;

var\_c dq ?;

var\_d dq ?;

iter dw 1;

iterator dw -1;

.code

start:

finit

startLoop:

xor ebx, ebx;

mov bx, iterator;

inc bx;

fld arr\_a[ebx \* 8];

fstp var\_a;

fld arr\_b[ebx \* 8];

fstp var\_b;

fld arr\_c[ebx \* 8];

fstp var\_c;

fld arr\_d[ebx \* 8];

fstp var\_d;

push offset var\_a

push offset var\_b

push offset var\_c

push offset var\_d

call WithPoint

inc iter;

inc iterator;

cmp iterator, 4

jnz startLoop

invoke ExitProcess, NULL;

end start

**Перевірка результатів розрахунків**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | №1 | №2 | №3 |
|  | | a=0.78  b=10.02  c=5.5  d=4.01 | a=1.78  b=10.03  c=4.5  d=5.01 | a=0.75  b=10.02  c=4.5  d=4.01 |
| 3\*a/2 | Власні розрахунки | 1.17 | 2.67 | 1.125 |
|  | Програма | 1.17 | 2.67 | 1.125 |
| ln(3\*a/2) | Власні розрахунки | 0.1570037488096651 | 0.98207847241215801 | 0.11778303565638303 |
|  | Програма | 0.157003748809665 | 0.982078472412158 | 0.117783035656383 |
|  | Власні розрахунки | 1.5285748211288669 | 1.8802820652265289 | 1.2399775493720939 |
|  | Програма | 1.528574821128867 | 1.880282065226529 | 1.239977549372094 |
|  | Власні розрахунки | 9.02 | 9.0300000000000000 | 9.02 |
|  | Програма | 9.02 | 9.029999999999999 | 9.02 |
|  | Власні розрахунки | -3.52 | -4.530000000000000 | -4.52 |
|  | Програма | -3.52 | -4.529999999999999 | -4.52 |
|  | Власні розрахунки | -0.434254202 | -0.41507331 | -0.274331301 |
|  | Програма | -0.4342542 | -0.4150733 | -0.2743313 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | №4 | №5 |
| a=1.75  b=-0.78  c=5.5  d=15.01 | a=1.75  b=-0.78  c=3.5  d=0.51 |
| 3\*a/2 | Власні розрахунки | 2.625 | 2.625 |
| Програма | 2.625 | 2.625 |
| ln(3\*a/2) | Власні розрахунки | 0.96508089604358702 | 0.96508089604358702 |
| Програма | 0.965080896043587 | 0.965080896043587 |
|  | Власні розрахунки | 1.3315032811202029 | 7.8278259940828031 |
| Програма | 1.331503281120203 | 7.827825994082803 |
|  | Власні розрахунки | -1.78 | -1.78 |
| Програма | -1.78 | -1.78 |
|  | Власні розрахунки | 7.28 | 5.28 |
| Програма | 7.28 | 5.28 |
|  | Власні розрахунки | 0.18289881 | 1.4825432 |
| Програма | 0.1828988 | 1.482543 |

**Висновки**

У ході виконання лабораторної роботи я вивчив прийоми розробки і використання процедур, представлених у вигляді динамічних бібліотек.

Я створив програму на мові асемблер, в якій згідно з індивідуальним варіантом завдання до лабораторної роботи 6 розробити процедуру на мові Асемблер, в якій реалізував обчислення заданого арифметичного виразу і виведення результатів у віконному інтерфейсі. Передаю значення змінних до процедури в якості параметрів. Процедуру розробив без вказівки та з використанням точки входу.

Розробив файл установок модуля DEF і виконав компіляцію обох варіантів файлу бібліотек DLL. Лістинги обох варіантів написання процедур, файлу DEF і командних рядків компіляції представив у звіті по лабораторній роботі.

Написав і скомпілював файл основної програми, в якому передбачив неявне завантаження розробленої DLL-бібліотеки з передачею в неї в циклі 5 значень змінних, заданих дійсними числами у вигляді одновимірних масивів.

Перевірив роботу основної програми з підключенням по черзі обох варіантів DLL- бібліотек і привів у звіті по лабораторній роботі лістинг основної програми і скріншоти вікон з результатами обчислень.

Написав і скомпілював файл основної програми, в якому передбачив явне завантаження розробленої DLL- бібліотеки з передачею до неї в циклі тих же 5 значень змінних.

Переконався в працездатності основної програми при зверненні в ній по черзі до кожної з DLL- бібліотек. Лістинг основної програми привів у звіті по лабораторній роботі.

Перевірив правильність виконання розрахунків і результатів, що виводяться, заздалегідь виконав контрольні розрахунки. Виконав відладку програми шляхом порівняння розрахованих програмою результатів з контрольними прикладами.

Програма здатна видавати повідомлення про помилку, якщо аргумент не належить до області визначення функції.

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.