**Лабораторная № 2**

***Цель.*** Получить навыки написания программ на языке высокого уровня со вставками на ассемблере.

***Задача.***

Написать на языке ассемблер программу, позволяющую следующее.

1. Ввести (взять из файла) несколько переменных.

2. Вычислить формулу , подставляя в неё введённые переменные.

Вывести результат на экран (в файл). Ввести (взять из файла) несколько переменных;

3. Вычислить формулу подставляя в неё введённые переменные;

Вывести результат на экран (в файл).

Программа скомпилирована компилятором link из пакета masm32.

Результат работы программы представлен на рис. 1.

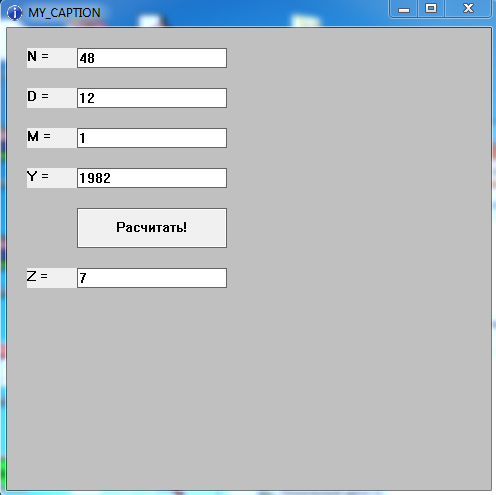


Рисунок 1

**Исходный код.**

Файл MODULE\_ONE:

.386

.model flat,stdcall

option casemap:none

include C:\masm32\INCLUDE\WINDOWS.INC

include C:\masm32\INCLUDE\KERNEL32.INC

include C:\masm32\INCLUDE\USER32.INC

include C:\masm32\INCLUDE\ADVAPI32.INC

include C:\masm32\INCLUDE\GDI32.INC

include my.inc

includelib C:\masm32\lib\comctl32.lib

includelib C:\masm32\lib\user32.lib

includelib C:\masm32\lib\gdi32.lib

includelib C:\masm32\lib\kernel32.lib

includelib C:\masm32\lib\user32.lib

includelib C:\masm32\lib\advapi32.lib

;###########################################################

MODULE\_CREATE\_CHILD\_proc PROTO hWnd1 :DWORD

;###########################################################

EXTERN HINST:DWORD

EXTERN HWNDN:DWORD

EXTERN HWNDM:DWORD

EXTERN HWNDD:DWORD

EXTERN HWNDY:DWORD

EXTERN HWNDZ:DWORD

;###########################################################

;data--data--data--data--data--data--data--data--data--data-- PROC

;----------------------------------------------------------------------------------------------

.DATA

EDIT\_CLASS DB "EDIT",0

String\_NULL DB 0 , 0 , 0 , 0

BUTTON\_CLASS DB "BUTTON",0

BUTTON\_NAME\_1 DB "Расчитать!",0

LABEL\_CLASS DB "STATIC",0

LABEL\_NAME\_N DB "N = ",0

LABEL\_NAME\_M DB "M = ",0

LABEL\_NAME\_D DB "D = ",0

LABEL\_NAME\_Y DB "Y = ",0

LABEL\_NAME\_Z DB "Z = ",0

;-

Green\_BRUSH DWORD Null

Red\_BRUSH DWORD Null

;##############################################################

;code--code--code--code--code--code--code--code--code--code-- PROC

;--------------------------------------------------------------------------------------------------

.CODE

START:

;---------------------------------------------------------------------------------------------------------------

; CREATE CHILD

;---------------------------------------------------------------------------------------------------------------

MODULE\_CREATE\_CHILD\_proc PROC hWnd1 :DWORD

local \_hWndGroup :DWORD

invoke CreateSolidBrush , 0033FF67h

mov Green\_BRUSH , EAX

;-1 StaticN

invoke CreateWindowEx , NULL , addr LABEL\_CLASS , \

addr LABEL\_NAME\_N , \

WS\_CHILD + WS\_VISIBLE , 20 ,20 , 50 ,20 , \

hWnd1 , NULL , HINST , NULL

;

;-2 StaticD

invoke CreateWindowEx , NULL , addr LABEL\_CLASS , \

addr LABEL\_NAME\_D , \

WS\_CHILD + WS\_VISIBLE , 20 ,60 , 50 ,20 , \

hWnd1 , NULL , HINST , NULL

;

;-3 StaticM

invoke CreateWindowEx , NULL , addr LABEL\_CLASS , \

addr LABEL\_NAME\_M , \

WS\_CHILD + WS\_VISIBLE , 20 ,100 , 50 ,20 , \

hWnd1 , NULL , HINST , NULL

;

;-4 StaticY

invoke CreateWindowEx , NULL , addr LABEL\_CLASS , \

addr LABEL\_NAME\_Y , \

WS\_CHILD + WS\_VISIBLE , 20 ,140 , 50 ,20 , \

hWnd1 , NULL , HINST , NULL

;

;-5 StaticZ

invoke CreateWindowEx , NULL , addr LABEL\_CLASS , \

addr LABEL\_NAME\_Z , \

WS\_CHILD + WS\_VISIBLE , 20 ,240 , 50 ,20 , \

hWnd1 , NULL , HINST , NULL

;

;-1 EditN

invoke CreateWindowEx , NULL , addr EDIT\_CLASS , \

addr String\_NULL , \

WS\_CHILD + WS\_VISIBLE + WS\_BORDER , 70 ,20 , 150 ,20 , \

hWnd1 , 1 , HINST , NULL

mov HWNDN, eax

;

;-2 EditD

invoke CreateWindowEx , NULL , addr EDIT\_CLASS , \

addr String\_NULL , \

WS\_CHILD + WS\_VISIBLE + WS\_BORDER , 70 ,60 , 150 ,20 , \

hWnd1 , 3 , HINST , NULL

mov HWNDD, eax

;

;-3 EditM

invoke CreateWindowEx , NULL , addr EDIT\_CLASS , \

addr String\_NULL , \

WS\_CHILD + WS\_VISIBLE + WS\_BORDER , 70 ,100 , 150 ,20 , \

hWnd1 , 2 , HINST , NULL

mov HWNDM, eax

;

;-4 EditY

invoke CreateWindowEx , NULL , addr EDIT\_CLASS , \

addr String\_NULL , \

WS\_CHILD + WS\_VISIBLE + WS\_BORDER , 70 ,140 , 150 ,20 , \

hWnd1 , 4 , HINST , NULL

mov HWNDY, eax

;

;-5 EditD

invoke CreateWindowEx , NULL , addr EDIT\_CLASS , \

addr String\_NULL , \

WS\_CHILD + WS\_VISIBLE + WS\_BORDER , 70 ,240 , 150 ,20 , \

hWnd1 , 5 , HINST , NULL

mov HWNDZ, eax

;

;-Button

invoke CreateWindowEx , NULL , addr BUTTON\_CLASS , \

addr BUTTON\_NAME\_1 , \

WS\_CHILD + WS\_VISIBLE + BS\_FLAT , 70 , 180 , 150 ,40 , \

hWnd1 , 7 , HINST , NULL

;

;-

RET 4

MODULE\_CREATE\_CHILD\_proc ENDP

;-------------------------------------------------------------------------------------------------------------------------------

; SEND MESSAGE to EDIT

;-------------------------------------------------------------------------------------------------------------------------------

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

END

Файл Step\_One:

.386

.model flat,stdcall

option casemap:none

include C:\masm32\INCLUDE\WINDOWS.INC

include C:\masm32\INCLUDE\KERNEL32.INC

include C:\masm32\INCLUDE\USER32.INC

include C:\masm32\INCLUDE\ADVAPI32.INC

include C:\masm32\INCLUDE\GDI32.INC

include my.inc

includelib C:\masm32\lib\comctl32.lib

includelib C:\masm32\lib\user32.lib

includelib C:\masm32\lib\gdi32.lib

includelib C:\masm32\lib\kernel32.lib

includelib C:\masm32\lib\user32.lib

includelib C:\masm32\lib\advapi32.lib

;###########################################################

MAIN\_WINDOW\_PROC PROTO :DWORD , :DWORD , :DWORD , :DWORD

MODULE\_CREATE\_CHILD\_proc PROTO hWnd1 :DWORD

IntegerIn PROTO : DWORD

;###########################################################

public HINST

public HWNDN

public HWNDM

public HWNDD

public HWNDY

public HWNDZ

;###########################################################

;data--data--data--data--data--data--data--data--data--data-- PROC

;----------------------------------------------------------------------------------------------

.DATA

HINST DWORD NULL

HWND\_WIN DWORD NULL

HWNDN DWORD NULL

HWNDM DWORD NULL

HWNDD DWORD NULL

HWNDY DWORD NULL

HWNDZ DWORD NULL

;text db 255,?,255 dup(0)

textN db 255 dup(0)

textM db 255 dup(0)

textD db 255 dup(0)

textY db 255 dup(0)

textZ db 255 dup(0)

rez db 255 dup(0)

n dw 0

m dw 0

d dw 0

y dw 0

z dw 0

a dw 0

b dw 0

x dw 0

;-

String\_CLASS DB "MY\_WINDOW",0

String\_CAPTION DB "MY\_CAPTION",0

OUTPUT DB 255 dup(0)

CONTROL DB "%d",0

;-

MSG\_WIN MSG <0>

;-

CONTENER DB 256 dup (0)

;##############################################################

;code--code--code--code--code--code--code--code--code--code-- PROC

;--------------------------------------------------------------------------------------------------

.CODE

START:

invoke GetModuleHandle , Null

mov HINST , EAX

;-

CALL MY\_REGISTER\_CLASS

cmp EAX , null

je EXIT

;-

;функия создания главного окна

invoke CreateWindowEx , NULL , addr String\_CLASS , addr String\_CAPTION , \

WS\_OVERLAPPEDWINDOW , 100 , 100 , 500 , 500 , \

NULL , NULL , HINST , NULL

;-

MOV HWND\_WIN , EAX

;-

invoke ShowWindow , HWND\_WIN , TRUE

invoke UpdateWindow , HWND\_WIN

;====================

;функция приёма сообщений прилоения

MSG\_LOOP:

invoke GetMessage , addr MSG\_WIN , null , null , null

CMP Eax , FALSE

JE EXIT

invoke TranslateMessage , addr MSG\_WIN

invoke DispatchMessage , addr MSG\_WIN

JMP MSG\_LOOP

;====================

EXIT:

invoke ExitProcess , Null

;----------------------------------------------------------------------------------------------------------

; GET POINT SCREEN

;----------------------------------------------------------------------------------------------------------

MY\_REGISTER\_CLASS PROC

local \_Struct\_WNDCLASS : WNDCLASS

;-

Mov \_Struct\_WNDCLASS.style , CS\_DBLCLKS ; стиль окна

; Mov Eax ,

Mov \_Struct\_WNDCLASS.lpfnWndProc , MAIN\_WINDOW\_PROC ; процедура окна

Mov \_Struct\_WNDCLASS.cbClsExtra , null ; дополнительная память для класса

Mov \_Struct\_WNDCLASS.cbWndExtra , null ; дополнительная память для окна

Mov Eax , HINST

Mov \_Struct\_WNDCLASS.hInstance , Eax ; handle приложения

Mov \_Struct\_WNDCLASS.lpszMenuName , NULL ; идентификатор меню

Mov \_Struct\_WNDCLASS.lpszClassName , offset String\_CLASS ; адрес строки класса

;-

invoke LoadIcon , NULL , IDI\_ASTERISK

Mov \_Struct\_WNDCLASS.hIcon , Eax

;-

invoke LoadCursor , NULL , IDC\_IBEAM

Mov \_Struct\_WNDCLASS.hCursor , Eax

;-

invoke CreateSolidBrush , 00FF0000h ; возвратит идентификатор кисти

invoke GetStockObject , LTGRAY\_BRUSH

Mov \_Struct\_WNDCLASS.hbrBackground , Eax

;==============

invoke RegisterClassA , addr \_Struct\_WNDCLASS

;==============

ret

MY\_REGISTER\_CLASS ENDP

;---------------------------------------------------------------------------------------------------------------------------

; WINDOW PROCEDURE

;---------------------------------------------------------------------------------------------------------------------------

MAIN\_WINDOW\_PROC PROC USES EBX ESI EDI \

hWnd\_ :DWORD , MESG :DWORD , wParam :DWORD , lParam:DWORD

LOCAL \_hwnd\_Win :DWORD

;-

CMP MESG , WM\_CREATE ;создание

JE WMCREATE

CMP MESG , WM\_COMMAND ;нажатие

JE BUTTON\_CHECK

CMP MESG , WM\_DESTROY ;закрытие

JE WMDESTROY

;----

DEF\_:

invoke DefWindowProc , hWnd\_ , MESG , wParam , lParam

jmp FINISH

;----

WMCREATE: ;создание дочерних компонентов

INVOKE MODULE\_CREATE\_CHILD\_proc , hWnd\_

jmp FINISH

BUTTON\_CHECK:

mov eax, wParam

cmp wParam, 0

je FINISH

cmp ax, 7

jne FINISH

shr eax, 16

cmp ax, BN\_CLICKED

jne FINISH

invoke SendMessage,[HWNDN],WM\_GETTEXT,10,addr textN ;вводим N

invoke SendMessage,[HWNDM],WM\_GETTEXT,10,addr textM ;вводим M

invoke SendMessage,[HWNDD],WM\_GETTEXT,10,addr textD ;вводим D

invoke SendMessage,[HWNDY],WM\_GETTEXT,10,addr textY ;вводим Y

;загрузка из эдиторов в переменные

lea eax, textN

invoke IntegerIn, eax

mov n, ax

lea eax, textM

invoke IntegerIn, eax

mov m, ax

lea eax, textD

invoke IntegerIn, eax

mov d, ax

lea eax, textY

invoke IntegerIn, eax

mov y, ax

;далее расчёт по формуле

formula:

cmp d, 15 ;

jg Dgreate ;если d>15, тогда на Dgreate

mov ax, m ;m в ax

imul n ;m\*n

mov a, ax ;результат в a

mov ax, y ;y в ax

idiv n ;y/n

mov b, ax ;результат в b

mov ax, a ;m\*n в ax

sub ax, [b] ;m\*n-y/b

mov [z], ax ;Z

jmp finish ;переход на метку finish

Dgreate:

mov ax, y ;y в ax

sub ax, m ;(y-m)

mov a, ax ;результат в a

mov ax, y ;y в ax

sub ax, d ;(y-d)

mov b, ax ;результат в b

mov ax, a ;(y-m) в ax

imul [b] ;(y-m)\*(y-d)

mov [z], ax ;Z

finish:

mov ax, z

invoke wsprintf, addr rez,addr CONTROL, ax ;перевод числа в строку

invoke SendMessage,[HWNDZ],WM\_SETTEXT,NULL,addr rez ;вывод результата

jmp FINISH

WMDESTROY:

invoke PostQuitMessage , False

jmp FINISH

;-

FINISH:

RET 16

MAIN\_WINDOW\_PROC ENDP

;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;===========================

IntegerIn proc text: DWORD

mov x,0

mov ebx, 0

mov eax, 0

mov edx, 0

mov ecx, 0 ;обнуляем счётчик

mov eax,10 ;умножаем на 10 каждый цикл

mov ebx, text ;первый символ

mov dl, [ebx] ;заносим в [dl]

cmp dl, '-' ;если -

je GetInteger ;число целое

loopGetUnsigned:

mov dl, [ebx] ;символ в dl

cmp dl, 0

je ff

mul x ;умножаем на 10

mov x, ax ;запоминаем результат

mov dl, [ebx] ;заносим в [dl]

sub dl, 30h ;получаем число

add x, dx ;прибавляем к результату

inc ebx ;переходим к следующему числу

mov ax,10 ;восстанавливаем множитель

jmp loopGetUnsigned

ff:

mov ax, x ;результат в ax

ret

GetInteger:

mov ax,10 ;умножаем на 10 каждый цикл

mov ch, 0 ;обнуляем счётчик

mov dh, 0 ;в dl будем заносить числа в [bx]

;dec cl ;пропускаем '-'

inc bx ;переходим к следующему числу

loopGetInteger:

mov dl, [ebx] ;символ в dl

cmp dl, 0

je ff2

mul x ;умножаем на 10

mov x, ax ;запоминаем результат

mov dl, [ebx] ;символ в dl

sub dl, 30h ;получаем число

add x, dx ;прибавляем к результату

inc ebx ;переходим к следующему числу

mov ax,10 ;восстанавливаем множитель

jmp loopGetInteger

ff2:

mov ax, x ;результат в ax

neg ax ;меняем знак

ret

IntegerIn endp

;===========================

END START