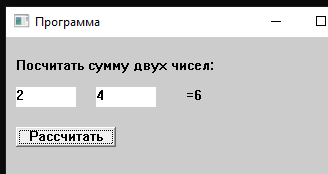
Лабораторная работа

Вычисление математической функции в конном приложении.

**Цель**: научиться создавать простейшее приложение с графическим интерфейсом средствами WinAPI на языке ассемблера, научиться вычислять математические функции на FPU на языке ассемблера.

В работе требуется реализовать оконное приложение выполняющее вычисление функции на FPU согласно заданию (варианты задания в конце описания лабораторной работы).

Внешний вид окна для вычисления суммы двух чисел приведен на рисунке



При вычислении функции окно должно иметь похожий вид, но вместо двух полей ввода иметь одно поле (аргумент для функции).

Для создания окна потребуется много типовых констант управляющих стилем и видом окна при создании. Константы, типы, функции, а также необходимые файлы для подключения объявлены в файлах win.inc и console.inc. Файлы представлены в папке с методичкой и должны быть подключены в проекте.

Пример подключения и константа определяющая цвет окна.

.586

.MODEL FLAT, stdcall

RGBW equ 00CCCCCCh ; цвет фона окна

include win.inc

include console.inc

# Создание окна

**Классы окон**

*Класс окна* определяет набор поведений, которые могут встречаться в нескольких окнах. Например, в группе кнопок каждая кнопка имеет аналогичное поведение, когда пользователь нажимает кнопку. Конечно, кнопки не полностью идентичны; Каждая кнопка отображает собственную текстовую строку и имеет собственные экранные координаты. Данные, уникальные для каждого окна, называются *данными экземпляра*.

Каждое окно должно быть связано с классом окна, даже если программа создает только один экземпляр этого класса. Важно понимать, что класс окна не является "классом" в смысле C++. Вместо этого это структура данных, используемая внутри операционной системы. Классы окон регистрируются в системе во время выполнения. Чтобы зарегистрировать новый класс окна, начните с заполнения структуры WNDCLASS:

C++

// Register the window class.

const wchar\_t CLASS\_NAME[] = L"Sample Window Class";

WNDCLASS wc = { };

wc.lpfnWndProc = WindowProc;

wc.hInstance = hInstance;

wc.lpszClassName = CLASS\_NAME;

Необходимо задать следующие члены структуры:

* WindowProc — это указатель на определяемую приложением функцию, называемую *процедурой окна* или «процедурой окна». Процедура окна определяет большую часть поведения окна. Далее мы рассмотрим процедуру Windows. Сейчас просто рассматривайте это как прямую ссылку.
* hInstance — это обработчик экземпляра приложения. Получите это значение из параметра *HINSTANCE* объекта **wWinMain**.
* CLASS\_NAME — это строка, идентифицирующая класс окна.

Имена классов являются локальными для текущего процесса, поэтому имя должно быть уникальным только в пределах процесса. Однако стандартные элементы управления Windows также имеют классы. При использовании любого из этих элементов управления необходимо выбрать имена классов, которые не конфликтуют с именами классов элементов управления. Например, класс окна для элемента управления Button называется "Button".

Структура WNDCLASS содержит другие элементы, не показанные здесь. Можно задать для них нулевое значение, как показано в этом примере, или заполнить их в. Эта структура подробно описана в документации по WNDCLASS.

Затем передайте адрес структуры WNDCLASS в функцию RegisterClass. Эта функция регистрирует класс окна в операционной системе.

C++

RegisterClass(&wc);

В ассемблере регистрация класса выполняется следующим образом

INVOKE RegisterClass, OFFSET WC

где WC WNDCLASS <?>

Но прежде структуру WC надо заполнить

Потребуются объявления

HINST DD 0 ; дескриптор приложения

CLASSNAME DB 'CLASS32',0

; заполнить структуру окна стиль

MOV WC.style, CS\_HREDRAW+CS\_VREDRAW+CS\_GLOBALCLASS

; процедура обработки сообщений

MOV WC.lpfnWndProc, OFFSET WNDPROC ;процедура обработки кода окна должна быть представлена в коде (пример процедуры будет приведен ниже)

MOV EAX, HINST

MOV WC.hInstance, EAX

INVOKE LoadIcon, 0, IDI\_APPLICATION ;грузит иконки

MOV WC.hIcon, EAX

INVOKE LoadCursor, 0, IDC\_ARROW ;грузит курсор

MOV WC.hCursor, EAX

INVOKE CreateSolidBrush, RGBW ;создает кисть для заполнения фона окна

MOV WC.hbrBackground, EAX

MOV DWORD PTR WC.lpszMenuName, 0

MOV DWORD PTR WC.lpszClassName, OFFSET CLASSNAME

Теперь можно регистрировать класс

INVOKE RegisterClass, OFFSET WC

**Создание окна**

Чтобы создать новый экземпляр окна, вызовите функцию [CreateWindowEx](https://docs.microsoft.com/ru-ru/windows/desktop/api/winuser/nf-winuser-createwindowexa) :

C++

HWND hwnd = CreateWindowEx(

0, // Optional window styles.

CLASS\_NAME, // Window class

L"Learn to Program Windows", // Window text

WS\_OVERLAPPEDWINDOW, // Window style

// Size and position

CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,

NULL, // Parent window

NULL, // Menu

hInstance, // Instance handle

NULL // Additional application data

);

if (hwnd == NULL)

{

return 0;

}

Подробное описание параметров можно прочитать в документации по функции [CreateWindowEx](https://docs.microsoft.com/ru-ru/windows/desktop/api/winuser/nf-winuser-createwindowexa) , но вот краткий обзор:

* Первый параметр позволяет указать некоторые необязательные поведения для окна (например, прозрачные окна). Присвойте этому параметру значение 0 для поведения по умолчанию.
* CLASS\_NAME имя класса окна. Определяет тип создаваемого окна.
* Текст окна используется различными способами для различных типов окон. Если окно содержит заголовок, текст отображается в заголовке окна.
* Стиль окна — это набор флагов, определяющих некоторый вид окна. Константа WS\_OVERLAPPEDWINDOW фактически является несколько флагов в сочетании с побитовой **или**. Вместе эти флаги предоставляют окну строку заголовка, границу, системное меню, а также кнопки **сворачивания** и **развертывания** . Этот набор флагов является наиболее распространенным стилем для окна приложения верхнего уровня.
* Для расположения и размера константа CW\_USEDEFAULT означает использование значений по умолчанию.
* Следующий параметр задает родительское окно или окно-владелец для нового окна. Если вы создаете дочернее окно, задайте родительский элемент. Для окна верхнего уровня задайте **значение NULL**.
* Для окна приложения следующий параметр определяет меню для окна. В этом примере не используется меню, поэтому значение равно **null**.
* *HINSTANCE* — это обработчик экземпляра, описанный выше. (См. [WinMain: точка входа приложения](https://docs.microsoft.com/ru-ru/windows/win32/learnwin32/winmain--the-application-entry-point).)
* Последний параметр является указателем на произвольные данные типа **void \***. Это значение можно использовать для передачи структуры данных в оконную процедуру. Мы покажем один из возможных способов использования этого параметра в разделе [Управление состоянием приложения](https://docs.microsoft.com/ru-ru/windows/win32/learnwin32/managing-application-state-).

[**CreateWindowEx**](https://docs.microsoft.com/ru-ru/windows/desktop/api/winuser/nf-winuser-createwindowexa) возвращает маркер в новое окно или нуль, если функция завершается ошибкой. Чтобы отобразить окно, т. е. сделать окно видимым, передайте в функцию [**ShowWindow**](https://docs.microsoft.com/ru-ru/windows/desktop/api/winuser/nf-winuser-showwindow) маркер окна:

C++

ShowWindow(hwnd, nCmdShow);

Параметр *HWND* — это дескриптор окна, возвращаемый функцией [**CreateWindowEx**](https://docs.microsoft.com/ru-ru/windows/desktop/api/winuser/nf-winuser-createwindowexa). Параметр *нкмдшов* можно использовать для сворачивания или разворачивания окна. Операционная система передает это значение в программу с помощью функции **wWinMain** .

Ниже приведен полный код для создания окна. Помните, что WindowProc все еще является прямым объявлением функции.

C++

// Register the window class.

const wchar\_t CLASS\_NAME[] = L"Sample Window Class";

WNDCLASS wc = { };

wc.lpfnWndProc = WindowProc;

wc.hInstance = hInstance;

wc.lpszClassName = CLASS\_NAME;

RegisterClass(&wc);

// Create the window.

HWND hwnd = CreateWindowEx(

0, // Optional window styles.

CLASS\_NAME, // Window class

L"Learn to Program Windows", // Window text

WS\_OVERLAPPEDWINDOW, // Window style

// Size and position

CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,

NULL, // Parent window

NULL, // Menu

hInstance, // Instance handle

NULL // Additional application data

);

if (hwnd == NULL)

{

return 0;

}

ShowWindow(hwnd, nCmdShow);

Поздравляем, вы создали окно! Сейчас окно не содержит никакого содержимого или взаимодействует с пользователем. В реальном приложении GUI окно будет реагировать на события пользователя и операционной системы. В следующем разделе описывается, как сообщения в окне предоставляют такую сортировку.

На ассемблере

; создать окно зарегистрированного класса

INVOKE CreateWindowEx, 0,

OFFSET CLASSNAME,

OFFSET TITL,

WS\_CAPTION + WS\_SYSMENU + WS\_THICKFRAME + WS\_GROUP + WS\_TABSTOP,

100, ; X — координата левого верхнего угла

100, ; Y — координата левого верхнего угла

400, ; DX - ширина окна

450, ; DY — высота окна

0, 0, HINST,0

CMP EAX, 0 ; проверка на ошибку

JZ END\_LOOP

MOV HWND, EAX ; дескриптор окна

INVOKE ShowWindow, HWND, SW\_SHOWNORMAL ; показать созданное окно

INVOKE UpdateWindow, HWND ;перерисовать видимую часть окна

Для окна примера сложения двух чисел потребуются следующие объявления

.data

HWND DD 0 ; дескриптор главного окна!

HINST DD 0 ; дескриптор приложения !

TITL DB "Программа",0

CLASSNAME DB 'CLASS32',0 !

Message MSG <?>

WC WNDCLASS <?> !

CPBUT db 'Рассчитать',0

CLSBTN db 'BUTTON',0

CLSEDT db 'EDIT',0

CAP db 'Сообщение',0

TEXTA db 20 dup(0) ; текст в полях редактирования

TEXTB db 20 dup(0)

summa dd 0

hBut DD ? ; дескриптор кнопки

hedt1 DD ? ; дескриптор поля 1

hedt2 DD ? ; дескриптор поля 2

hdc DD ? ; дескриптор контекста окна

ps PAINTSTRUCT <?>

mess1 db 'Посчитать сумму двух чисел: ',0 ; надпись в окне

mess1\_len equ $-mess1-1

mess2 db '=',10 dup(' '),0 ; результат суммы строковый

sum\_len equ $-mess2-1

Исполняемый код начинается с определения дескриптора исполняемого приложения, для которого будем создавать окно

.code

START proc

; получить дескриптор приложения

INVOKE GetModuleHandle, 0

MOV HINST, EAX

Далее необходимо

;заполнить структуру окна стиль

…

;указать процедуру обработки сообщений

…

;создать окно зарегистрированного класса

А также вывести окно на экран

INVOKE ShowWindow, HWND, SW\_SHOWNORMAL ; показать созданное окно

INVOKE UpdateWindow, HWND ;перерисовать видимую часть окна

Но помимо окна потребуется еще создать цикл обработки сообщений системы и процедуру обработки событий для окна

Все элементы в windows обмениваются сообщениями, поэтому требуется код для приема и диспетчеризации сообщений

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

; цикл обработки сообщений

MSG\_LOOP:

INVOKE GetMessage, OFFSET Message, 0,0,0

CMP EAX, 0

JE END\_LOOP

INVOKE TranslateMessage, OFFSET Message

INVOKE DispatchMessageA, OFFSET Message

JMP MSG\_LOOP

END\_LOOP:

INVOKE ExitProcess, Message.wParam ; выход из программы

START endp

Также требуется основной цикл оконного приложения который будет реагировать на полученные сообщения и вызывать требуемые процедуры обработки

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

; процедура окна

WNDPROC PROC hW:DWORD, Mes:DWORD, wParam:DWORD, lParam:DWORD

CMP Mes, WM\_DESTROY ;будет выполняться при закрытии окна

JE WMDESTROY

CMP Mes, WM\_CREATE ; при создании окна

JE WMCREATE

CMP Mes, WM\_COMMAND ; общие команды (сообщения)

JE WMCOMMAND

CMP Mes, WM\_PAINT ; события перерисовки формы окна

JE WMPAINT

JMP DEFWNDPROC

И собственно функции обработки данных событий

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

WMCREATE: ; создание окна

INVOKE CreateWindowExA, 0, ; поле редактирования 1

offset CLSEDT, ; имя класса окна

offset TEXTA, ; надпись в поле

WS\_CHILD+WS\_VISIBLE, ; стиль окна

10, 50, ;x, y

60, 20, ; длина, ширина

hW, ;дескриптор окна

0, ;дескриптор меню

HINST, ;дескриптор приложения

0 ;lpParam

mov hedt1,eax ; сохранение дескриптора

mov eax,0

INVOKE ShowWindow, hedt1, SW\_SHOWNORMAL

INVOKE CreateWindowExA, 0, ; поле редактирования 2

offset CLSEDT, ; имя класса окна

offset TEXTB, ; надпись в поле

WS\_CHILD+WS\_VISIBLE, ; стиль окна

90, 50, ;x, y

60, 20, ; длина, ширина

hW, ;дескриптор окна

0, ;дескриптор меню

HINST, ;дескриптор приложения

0 ;lpParam

mov hedt2,eax ; сохранение дескриптора

mov eax,0

INVOKE ShowWindow, hedt2, SW\_SHOWNORMAL

INVOKE CreateWindowExA, 0, ; кнопка

offset CLSBTN, ; имя класса окна

offset CPBUT, ; надпись на кнопке

WS\_CHILD+WS\_VISIBLE, ; стиль окна кнопки

10, 90, ;x, y

100, 20, ; длина, ширина

hW, ;дескриптор окна

0, ;дескриптор меню

HINST, ;дескриптор приложения

0 ;lpParam

mov hBut,eax ; сохранение дескриптора

mov eax,0

INVOKE ShowWindow, hBut, SW\_SHOWNORMAL

MOV EAX, 0

JMP FINISH

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

WMCOMMAND: ; обработка нажатия кнопки

mov eax, hBut

cmp lParam,eax

jne COM\_END ; команда не соответствует нажатию кнопки

INVOKE SendMessage, hedt1, WM\_GETTEXT, 20, offset TEXTA

INVOKE SendMessage, hedt2, WM\_GETTEXT, 20, offset TEXTB

INVOKE StrToInt, offset TEXTA

mov summa, eax

INVOKE StrToInt, offset TEXTB

add summa, eax

mov eax, sum\_len

INVOKE TextOutA, ; стирание строки результата в окне

hdc, 180, 50, offset mess2, eax

INVOKE IntToStr, summa, offset mess2+1

INVOKE LENSTR, offset mess2 ; определение длины результата

push eax

INVOKE TextOutA, ; вывод результата

hdc, 180, 50, offset mess2, eax

pop ecx ; очистка строки

inc ecx

mov al,' '

mov edi, offset mess2+1

CLR: mov [edi],al

inc edi

loop CLR

COM\_END:

MOV EAX, 0

JMP FINISH

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

WMPAINT: ; перерисовка окна

INVOKE BeginPaint, hW, offset ps

mov hdc,eax

INVOKE SetBkColor, hdc, RGBW

mov eax, mess1\_len

INVOKE TextOutA, hdc, 10, 20, offset mess1, eax

INVOKE EndPaint, hdc, offset ps

MOV EAX, 0

JMP FINISH

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

DEFWNDPROC: ; обработка сообщения по умолчанию

INVOKE DefWindowProc,

hW, Mes, wParam, lParam

JMP FINISH

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

WMDESTROY: ; выход из цикла обработки сообщений

INVOKE PostQuitMessage, 0

MOV EAX, 0

FINISH: ret

WNDPROC ENDP

END START

Итак, приведен пример создания окна для сложения двух чисел.

**Поэкспериментируйте с стилями окна и измените внешний вид окна.**

**Задание**

**Запрограммируйте функцию согласно варианту (№ варианта = номеру бригады)**

**Вариант1 – cos (2\*x)**

**Вариант2 – sin (5\*x)**

**Вариант3 – sin(x)\*cos(x)**

**Вариант4 – tan (x)**

**Вариант5 – 1/tan(x)**

**Вариант6 – log (2\*x)**

**Вариант7 – ln (x)**

**Вариант8 – sin(x)/cos(x)**

**Вариант9 – arctan (x)**

**Вариант10 – arcsin(x)**

**Вариант11 – arccos(x)**

**Вариант12 – sin(pi\*x)**

**Инструкции FPU подробно изложены в документе FPU.pdf**