https://nestor.minsk.by/kg/2008/22/kg82215.html

[https://nestor.minsk.by/kg/logo.jpg](https://nestor.minsk.by/kg/)

[](https://nestorclub.com/index.pl?act=NEWSSHOW&id=2020120301)

**Ассемблер под Windows для чайников. Часть 6**

[главная](https://nestor.minsk.by/kg/) » [статьи](https://nestor.minsk.by/kg/arch.html) » [2008](https://nestor.minsk.by/kg/2008/) » [22](https://nestor.minsk.by/kg/2008/22/) » 06.06.2008‣программирование

**Итак, дорогие друзья, вы уже знаете многое, но от этого область вашего невежества становится лишь все больше и больше. Так уж устроен мир: пока мы не задумываемся о принципах существования или устройства того или иного объекта, мы имеем пробел лишь в одной области — области этого объекта, однако стоит лишь начать изучать этот объект, как оказывается, что список связанных с ним неизвестных нам доселе объектов уходит за горизонт нашего сознания.**

Для тех, кто решительно настроен покорять эти горизонты в мире программирования под Windows, я продолжаю свой рассказ про FASM. Сегодня мы закрываем тему ресурсов и приступаем к созданию собственного текстового редактора. Именно поэтому в прошлой статье я просил вас рассмотреть самостоятельно ресурсы в примере MINIPAD из папки EXAMPLES. Если у вас с этим возникли затруднения — попробуйте разобраться со второго захода, теперь уже с моими пояснениями:

**; Sampl text-editor**

**format PE GUI 4.0**

**entry start**

**include 'win32a.inc'**

**include 'encoding\WIN1251.INC'**

**section '.data' data readable writeable**

**title db 'Мой Первый Текстовый Редактор',0**

**class db 'FASMWIN32',0**

**\_edit db 'EDIT',0**

**nofnc db 'Эта функция недоступна в демо-версии',0**

**errtxt db 'Код ошибки: %u',0**

**errbuf rb $-errtxt+10**

**hwnd dd ?**

**hmenu dd ?**

**hedit dd ?**

**hacc dd ?**

**font dd ?**

**wc WNDCLASS 0,WindowProc,0,0,0,0,0,COLOR\_BTNFACE+1,0,class**

**msg MSG**

**client RECT**

**menuinfo MENUITEMINFO sizeof.MENUITEMINFO,MIIM\_STATE**

**section '.code' code readable executable**

**start:**

**invoke GetModuleHandle,0**

**mov [wc.hInstance],eax**

**invoke LoadIcon,[wc.hInstance],IDI\_MAIN**

**mov [wc.hIcon],eax**

**invoke LoadCursor,0,IDC\_ARROW**

**mov [wc.hCursor],eax**

**invoke RegisterClass,wc**

**cmp eax,0**

**je error**

**invoke LoadAccelerators,[wc.hInstance],IDA\_MAIN**

**mov [hacc],eax**

**invoke LoadMenu,[wc.hInstance],IDM\_MAIN**

**mov [hmenu],eax**

**invoke CreateWindowEx,0,class,title,WS\_VISIBLE+WS\_OVERLAPPEDWINDOW,\**

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

**0,eax,[wc.hInstance],0**

**cmp eax,0**

**je error**

**mov [hwnd],eax**

**msg\_loop:**

**invoke GetMessage,msg,0,0,0**

**cmp eax,0**

**je end\_loop**

**invoke TranslateAccelerator,[hwnd],[hacc],msg**

**cmp eax,0**

**jne msg\_loop**

**invoke TranslateMessage,msg**

**invoke DispatchMessage,msg**

**jmp msg\_loop**

**error:**

**invoke GetLastError**

**invoke wsprintf,errbuf,errtxt,eax**

**invoke MessageBox,0,errbuf,0,MB\_OK**

**end\_loop:**

**invoke ExitProcess,[msg.wParam]**

**proc WindowProc hwnd,wmsg,wparam,lparam**

**push ebx esi edi**

**cmp [wmsg],WM\_COMMAND**

**je .wmcommand**

**cmp [wmsg],WM\_CREATE**

**je .wmcreate**

**cmp [wmsg],WM\_SIZE**

**je .wmsize**

**cmp [wmsg],WM\_SETFOCUS**

**je .wmsetfocus**

**cmp [wmsg],WM\_DESTROY**

**je .wmdestroy**

**.defwndproc:**

**invoke DefWindowProc,[hwnd],[wmsg],[wparam],[lparam]**

**jmp .finish**

**.wmcommand:**

**mov eax,[wparam]**

**cmp ax,IDM\_NEW**

**je .NEW**

**cmp ax,IDM\_OPEN**

**je .OPEN**

**cmp ax,IDM\_SAVE**

**je .SAVE**

**cmp ax,IDM\_SAVEAS**

**je .SAVEAS**

**cmp ax,IDM\_EXIT**

**je .EXIT**

**cmp ax,IDM\_UNDO**

**je .UNDO**

**cmp ax,IDM\_CUT**

**je .CUT**

**cmp ax,IDM\_COPY**

**je .COPY**

**cmp ax,IDM\_PASTE**

**je .PASTE**

**cmp ax,IDM\_DELETE**

**je .DELETE**

**cmp ax,IDM\_SELECTALL**

**je .SELECTALL**

**cmp ax,IDM\_ABOUT**

**je .ABOUT**

**jmp .finish**

**; обработчики сообщений меню файл:**

**.NEW:**

**invoke MessageBox,0,nofnc,title,MB\_OK**

**jmp .finish**

**.OPEN:**

**invoke MessageBox,0,nofnc,title,MB\_OK**

**jmp .finish**

**.SAVE:**

**invoke MessageBox,0,nofnc,title,MB\_OK**

**jmp .finish**

**.SAVEAS:**

**invoke MessageBox,0,nofnc,title,MB\_OK**

**jmp .finish**

**.EXIT:**

**invoke DestroyWindow,[hwnd]**

**jmp .finish**

**; обработчики сообщений меню правка:**

**.UNDO:**

**mov eax,EM\_UNDO**

**jmp .send2editbox**

**.CUT:**

**mov eax,WM\_CUT**

**jmp .send2editbox**

**.COPY:**

**mov eax,WM\_COPY**

**jmp .send2editbox**

**.PASTE:**

**mov eax,WM\_PASTE**

**jmp .send2editbox**

**.DELETE:**

**mov eax,WM\_CLEAR**

**jmp .send2editbox**

**.send2editbox:**

**invoke SendMessage,[hedit],eax,0,0**

**jmp .finish**

**.SELECTALL:**

**invoke SendMessage,[hedit],EM\_SETSEL,0,-1**

**jmp .finish**

**.ABOUT:**

**invoke DialogBoxParam,[wc.hInstance],IDD\_ABOUT,[hwnd],AboutDialog,0**

**jmp .finish**

**.wmcreate:**

**invoke GetClientRect,[hwnd],client**

**invoke CreateWindowEx,WS\_EX\_CLIENTEDGE,\_edit,0,WS\_VISIBLE+WS\_CHILD+ WS\_HSCROLL+WS\_VSCROLL+ES\_AUTOHSCROLL+ES\_AUTOVSCROLL+ ES\_MULTILINE,[client.left],[client.top],[client.right],[client.bottom],[hwnd],0,[wc.hInstance],NULL**

**cmp eax,0**

**je .failed**

**mov [hedit],eax**

**invoke CreateFont,16,0,0,0,0,FALSE,FALSE,FALSE,RUSSIAN\_CHARSET, OUT\_RASTER\_PRECIS,CLIP\_DEFAULT\_PRECIS,DEFAULT\_QUALITY, FIXED\_PITCH+FF\_DONTCARE,NULL**

**cmp eax,0**

**je .failed**

**mov [font],eax**

**invoke SendMessage,[hedit],WM\_SETFONT,eax,FALSE**

**mov eax,0**

**jmp .finish**

**.failed:**

**mov eax,-1**

**jmp .finish**

**.wmsize:**

**invoke GetClientRect,[hwnd],client**

**invoke MoveWindow,[hedit],[client.left],[client.top],[client.right],[client.bottom],TRUE**

**mov eax,0**

**jmp .finish**

**.wmsetfocus:**

**invoke SetFocus,[hedit]**

**mov eax,0**

**jmp .finish**

**.wmdestroy:**

**invoke PostQuitMessage,0**

**mov eax,0**

**.finish:**

**pop edi esi ebx**

**ret**

**endp**

**proc AboutDialog hwnd,msg,wparam,lparam**

**push ebx esi edi**

**cmp [msg],WM\_COMMAND**

**je .close**

**cmp [msg],WM\_CLOSE**

**je .close**

**mov eax,0**

**jmp .finish**

**.close:**

**invoke EndDialog,[hwnd],0**

**.processed:**

**mov eax,1**

**.finish:**

**pop edi esi ebx**

**ret**

**endp**

**section '.idata' import data readable writeable**

**library kernel32,'KERNEL32.DLL',\**

**user32,'USER32.DLL',\**

**gdi32,'GDI32.DLL'**

**include 'api\kernel32.inc'**

**include 'api\user32.inc'**

**include 'api\gdi32.inc'**

**section '.rsrc' resource data readable**

**IDM\_MAIN = 101**

**IDA\_MAIN = 201**

**IDD\_ABOUT = 301**

**IDI\_MAIN = 401**

**IDM\_NEW = 1101**

**IDM\_OPEN = 1102**

**IDM\_SAVE = 1103**

**IDM\_SAVEAS = 1104**

**IDM\_EXIT = 1109**

**IDM\_UNDO = 1201**

**IDM\_CUT = 1202**

**IDM\_COPY = 1203**

**IDM\_PASTE = 1204**

**IDM\_DELETE = 1205**

**IDM\_SELECTALL = 1206**

**IDM\_ABOUT = 1401**

**directory RT\_MENU,menus,\**

**RT\_ACCELERATOR,accelerators,\**

**RT\_DIALOG,dialogs,\**

**RT\_GROUP\_ICON,group\_icons,\**

**RT\_ICON,icons,\**

**RT\_VERSION,versions**

**resource menus,\**

**IDM\_MAIN,LANG\_RUSSIAN+SUBLANG\_DEFAULT,main\_menu**

**resource accelerators,\**

**IDA\_MAIN,LANG\_ENGLISH+SUBLANG\_DEFAULT,main\_keys**

**resource dialogs,\**

**IDD\_ABOUT,LANG\_RUSSIAN+SUBLANG\_DEFAULT,about\_dialog**

**resource group\_icons,\**

**IDI\_MAIN,LANG\_NEUTRAL,main\_icon**

**resource icons,\**

**1,LANG\_NEUTRAL,main\_icon\_data**

**resource versions,\**

**1,LANG\_NEUTRAL,version**

**menu main\_menu**

**menuitem '&Файл',0,MFR\_POPUP**

**menuitem <'Созд&ать',9,'Ctrl+N'>,IDM\_NEW,0,MFS\_GRAYED**

**menuitem <'&Открыть…',9,'Ctrl+O'>,IDM\_OPEN,0,MFS\_GRAYED**

**menuitem <'&Сохранить',9,'Ctrl+S'>,IDM\_SAVE,0,MFS\_GRAYED**

**menuitem 'Сохранить &как…',IDM\_SAVEAS,0,MFS\_GRAYED**

**menuseparator**

**menuitem <'В&ыход',9,'Ctrl+Q'>,IDM\_EXIT,MFR\_END**

**menuitem '&Правка',0,MFR\_POPUP**

**menuitem <'&Отменить',9,'Ctrl+Z'>,IDM\_UNDO**

**menuseparator**

**menuitem <'&Вырезать',9,'Ctrl+X'>,IDM\_CUT**

**menuitem <'&Копировать',9,'Ctrl+C'>,IDM\_COPY**

**menuitem <'Вст&авить',9,'Ctrl+V'>,IDM\_PASTE**

**menuitem <'&Удалить',9,'Del'>,IDM\_DELETE**

**menuseparator**

**menuitem <'Выделить в&се',9,'Ctrl+A'>,IDM\_SELECTALL,MFR\_END**

**menuitem '&Вид',0**

**menuitem '&Справка',0,MFR\_POPUP+MFR\_END**

**menuitem '&О программе',IDM\_ABOUT,MFR\_END**

**accelerator main\_keys,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'N',IDM\_NEW,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'O',IDM\_OPEN,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'S',IDM\_SAVE,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'Q',IDM\_EXIT,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'Z',IDM\_UNDO,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'X',IDM\_CUT,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'C',IDM\_COPY,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'V',IDM\_PASTE,\**

**FVIRTKEY+FNOINVERT+FCONTROL,'A',IDM\_SELECTALL**

**dialog about\_dialog,'О программе',40,40,172,60,WS\_CAPTION+WS\_POPUP+WS\_SYSMENU+DS\_MODALFRAME**

**dialogitem 'STATIC',<'Мой Первый Текстовый Редактор',0Dh,0Ah,'Copyright ',0A9h,' BarMentaLisk 2008.'>,-1,27,10,144,40,WS\_VISIBLE+SS\_CENTER**

**dialogitem 'STATIC',IDI\_MAIN,-1,8,8,32,32,WS\_VISIBLE+SS\_ICON**

**dialogitem 'STATIC','',-1,4,34,164,11,WS\_VISIBLE+SS\_ETCHEDHORZ**

**dialogitem 'STATIC','Написан при помощи FASM',-1,12,42,100,20,WS\_VISIBLE+SS\_LEFT**

**dialogitem 'BUTTON','OK',IDOK,124,40,42,14,WS\_VISIBLE+ WS\_TABSTOP+BS\_DEFPUSHBUTTON**

**enddialog**

**icon main\_icon,main\_icon\_data,'1.ico'**

**versioninfo version,VOS\_NT\_WINDOWS32,VFT\_APP,VFT2\_UNKNOWN,LANG\_RUSSIAN+ SUBLANG\_DEFAULT,0,\**

**'Comments','Написан при помощи FASM',\**

**'CompanyName','BarMentaLisk',\**

**'FileDescription','Текстовый редактор',\**

**'ProductName',<'Мой Первый',0Dh,0Ah,'Текстовый Редактор'>,\**

**'LegalCopyright',<'Copyright ',0A9h, 'BarMentaLisk 2008'>,\**

**'FileVersion','0.1.0.0',\**

**'OriginalFilename','editor1.EXE'**

**; EOF**

Видали, сколько текста? Я бы на вашем месте не рискнул все это набирать вручную, а воспользовался бы электронной копией с форума. Хотя при вводе кода вручную он намного лучше понимается и запоминается. Действует такой же принцип, как и при конспектировании услышанного материала. В секции данных вам все должно быть понятно, кроме **rb $-errtxt+10**. У символа $ есть несколько значений в зависимости от контекста. К примеру, он может использоваться для обозначения шестнадцатеричного числа, как в паскале: $FACE8D. Он также является специальным символом, обозначающим текущее относительное смещение. Подробно о смещениях я планирую рассказать вам в следующих статьях, но сейчас тем из вас, кто вообще не имеет представления о смещениях, скажу "по-колхозному": смещение — это номер байта в программе, причем отсчет начинается с нулевого байта, затем идет первый, второй и так далее. Наши данные укладываются в секцию данных друг за дружкой в каждый байт. Мы знаем, что title является указателем на первый символ строки заголовка, а class — на первый символ строки класса, причем class идет следом за title. Значит, class-title=30, потому что ровно 30 байт занимают данные title (29 символов + завершающий ноль). Под буфер errbuf необходимо выделить ровно на 10 байт больше памяти, чем занимает строка errtxt. В таком случае мы могли бы использовать инструкцию rb 15+10 или rb 25. Но тогда каждый раз при изменении длины строки придется изменять и размер буфера. А если подобных строк и буферов много, то очень легко запутаться. Так что, если размер буфера относителен, то надежнее выражать его таким образом. $ — это относительный номер текущего байта — значит, $-errtxt (при условии, что выражение следует сразу за errtxt) при компиляции будет всегда замещаться размером этой строки в байтах. Такие вот дела. Напоминаю, что LoadIcon загружает указанную иконку из ресурсов исполняемого файла. Первый параметр — идентификатор исполняемого файла или ноль для загрузки стандартной иконки. Второй параметр — строка — имя иконки из ресурсов либо идентификатор стандартной иконки, если первый параметр ноль. Теперь мы указываем наше приложение в качестве первого параметра и идентификатор нашей иконки (для удобства используется псевдоним IDI\_MAIN=401) в качестве второго. Курсор оставляем стандартный. Меню и акселераторы ставим свои.  
  
Акселераторы — это горячие клавиши или комбинации клавиш для быстрого выполнения тех или иных действий в программе. Они также являются одним из типов ресурсов, тип — RT\_ACCELERATOR. Описываются макроинструкцией accelerator, за которой после имени ресурса тройками через запятую следуют параметры: флаги акселератора; код виртуальной клавиши или символ ASCII; идентификатор. Список флагов (Accelerator flags) и кодов виртуальных клавиш (Virtual key codes), как обычно, смотрим в EQUATES\USER32.INC. Атрибуты FALT, FCONTROL и FSHIFT означают, что вместе с указанной клавишей нажаты Alt, Ctrl, Shift или их комбинация. Атрибут FVIRTKEY означает, что событие происходит при нажатии соответствующей виртуальной клавиши (символ клавиши в кавычках должен быть в верхнем регистре). При отсутствии этого атрибута горячей клавишей будет считаться указанный символ — в этом случае Alt, Ctrl, Shift не учитываются, а также клавиша получится зависимой от регистра и раскладки. FNOINVERT означает, что выбранный при помощи акселератора пункт не будет подсвечиваться — этот атрибут является устаревшим и используется лишь для обратной совместимости с 16-битными версиями Windows. Функция LoadAccelerators используется аналогично функции LoadMenu, только возвращаемое ею значение — дескриптор загруженной таблицы акселераторов. В цикл обработки сообщений включена функция TranslateAccelerator. Она обрабатывает сообщения о нажатии горячих клавиш — акселераторов нашего меню, переводя их в сообщения WM\_COMMAND или WM\_SYSCOMMAND. Чтобы переведенные функцией сообщения можно было отличить от сообщений меню или элементов управления, старший байт параметра wparam сообщения устанавливается в единицу (табл. 1). Параметры: дескриптор обрабатываемого окна; дескриптор таблицы акселераторов; указатель на структуру MSG. При ошибке функция возвращает ноль. Если результат отличен от нуля — значит, сообщение обработано и не нуждается в дополнительной обработке функцией TranslateMessage.

|  |  |  |  |
| --- | --- | --- | --- |
| Сообщение от: | wParam (старшее слово) | wParam (младшее слово) | lParam |
| Меню | 0 | идентификатор меню (IDM\_\*) | 0 |
| Акселератора | 1 | идентификатор акселератора(IDM\_\*) | 0 |
| Элемента управления | событие | идентификатор элемента | дескриптор элемента (handle) |

Для дополнительной обработки ошибок использована функция GetLastError. Эта функция возвращает код последней ошибки (если была ошибка), произошедшей при вызове API-Функции. При помощи функции wsprintf мы приводим код ошибки к символьному виду и выводим на экран функцией MessageBox. Список кодов ошибок можно найти в MSDN или в каком-либо справочнике по Windows, хотя даже у самой Microsoft отсутствует полный список кодов ошибок. Обработчики сообщений меню файл за исключением пункта Выход в ресурсах установлены как недоступные (MFS\_GRAYED), но на случай их разблокировки каким-нибудь взломщиком обрабатываются выводом сообщения о недоступности функции. Обработку этих пунктов мы добавим и изучим в следующей статье. Выход обрабатывается вызовом функции DestroyWindow. Эта функция предназначена для корректного уничтожения окна со всеми его элементами, зависимыми окнами. Она посылает окну сообщения WM\_DESTROY и WM\_NCDESTROY. Единственный ее параметр — дескриптор уничтожаемого окна. При обработке пунктов "отменить", "вырезать", "копировать", "вставить", "удалить" производится отправка соответствующего сообщения окну редактирования. Параметры всех сообщений, кроме EM\_SETSEL (выделение текста), не используются и должны быть нулями. Параметры EM\_SETSEL: номер символа начала выделения; номер символа конца выделения. Если первый параметр равен нулю, а второй — минус единице, то выделяется весь текст в окне. Если первый параметр равен минус единице — все выделения снимаются.



Обработчик пункта О программе вызывает функцию DialogBoxParam. Эта функция создает диалоговое окно (dialog box) — окно, разработанное специально для быстрого и удобного создания окон с дочерними элементами. Все его элементы описываются в одном блоке ресурсов и автоматически создаются средствами операционной системы при создании окна. Параметры: идентификатор исполняемого модуля программы; идентификатор ресурса, содержащего шаблон диалогового окна; дескриптор окна-владельца; указатель на процедуру диалогового окна; значение, передаваемое диалоговому окну в параметре lParam сообщения WM\_INITDIALOG. Описание элементов окна в шаблоне интуитивно понятно и схоже с параметрами функции CreateWindowEx. Процедура диалогового окна аналогична процедуре обычного окна. Более подробно о диалоговых окнах мы поговорим в одной из следующих статей. Функция GetClientRect возвращает координаты клиентской области указанного окна в указанную структуру RECT. Параметры: дескриптор окна; указатель на структуру RECT. Причем в элементы структуры left и top возвращаются нули, а в элементы right и bottom — соответственно ширина и высота клиентской области окна. Эти данные необходимы нам, чтобы правильно задать размеры окна для редактирования. Нижеследующие команды имеют смысл лишь при успешном создании окна, поэтому, если в eax возвращается ноль (ошибка), мы пропускаем остальные команды. CreateFont создает или изменяет шрифт с указанными параметрами: высота шрифта, ноль для высоты по умолчанию; ширина шрифта, ноль для оптимальной пропорции; угол наклона текста в десятых долях градуса; угол наклона символов в десятых долях градуса; жирность шрифта (0-1000, 0 — по умолчанию, 400 — нормальный, 700 — жирный); false либо true = 0 либо 1 = обычный либо курсив; обычный либо подчеркнутый; обычный либо зачеркнутый; набор символов — например, ANSI\_CHARSET или DEFAULT\_CHARSET (список ищем в INCLUDE\EQUATES\GDI32.INC); флаг степени соответствия шрифта заданным параметрам — например, OUT\_DEFAULT\_PRECIS (список ищем там же); флаг степени сглаживания шрифта — например, CLIP\_DEFAULT\_PRECIS (список там же); флаг качества шрифта — например, DEFAULT\_QUALITY (список там же); сдвиг и семейство шрифта — например, DEFAULT\_PITCH+FF\_DONTCARE; указатель на строку с названием шрифта или ноль для оптимального выбора. Возвращаемое в eax значение — дескриптор созданного шрифта либо ноль при ошибке. Сообщение WM\_SETFONT устанавливает указанный шрифт. Параметры: дескриптор шрифта; флаг перерисовки окна с новым шрифтом (0 либо 1).  
  
При изменении размеров основного окна (сообщение WM\_SIZE) необходимо изменять размеры и дочернего окна редактирования. Для этого мы снова узнаем размер главного окна и перемещаем дочернее — MoveWindow. Эта функция изменяет позицию окна и его размеры. Параметры: дескриптор окна; позиция X; позиция Y; ширина; высота; перерисовки окна (0 либо 1).  
  
На сегодня, думаю, хватит. Продолжение, естественно, следует.  
  
Все приводимые примеры были протестированы на правильность работы под Windows XP и, скорее всего, будут работать под другими версиями Windows, однако я не даю никаких гарантий их правильной работы на вашем компьютере.