https://nestor.minsk.by/kg/2008/26/kg82612.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 7**

[**главная**](https://nestor.minsk.by/kg/)**»**[**статьи**](https://nestor.minsk.by/kg/arch.html)**»**[**2008**](https://nestor.minsk.by/kg/2008/)**»**[**26**](https://nestor.minsk.by/kg/2008/26/)**» 04.07.2008‣программирование**

**Заждались, наверное, продолжения? Что ж, ожидание только усиливает радость встречи. На сегодня у нас запланирован выход бета-версии 0.2 нашего первого текстового редактора. Он, к сожалению, по возможностям пока что немного уступает даже стандартному "Блокноту", но это ведь только во благо вашему обучению. В дальнейшем у вас будет шанс для усовершенствования программы в меру своих возможностей и, самое главное, знаний. Вы же не ждете, что я всю жизнь буду программировать за вас?**

Надеюсь, что нет, потому что основная цель этих занятий — развить интерес к программированию на ассемблере под Windows и изучить его основные моменты. Ну а время покажет, кому что ближе. Кого-то потянет в низкоуровневое программирование, кто-то, привязавшись к макросам, подастся в высокий уровень, забросив ассемблер… Но это все в будущем.  
  
А сейчас мы открываем привычный для нас FASMW.EXE и вводим код, электронную копию которого, как обычно, можно скачать на форуме. Код программы частично соответствует коду, приведенному в 6-й части цикла, но только частично. Так что если решите сэкономить время и скопировать набранные куски из прошлой версии — будьте предельно внимательны и сверяйте каждую строчку.

**format PE GUI 4.0**

**entry start**

**include 'win32a.inc'**

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

**MAXSIZE equ 260 ; максимальное имя файла в байтах**

**MEMSIZE equ 65537 ; размер временного буфера**

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

**title db 'Простой Текстовый Редактор. Vasylenko Roman (2021)v01',0**

**class db 'FASMWIN32',0**

**edit db 'EDIT',0**

**saveq db 'Сохранить изменения ?',0**

**filter db 'Text Files',0,'\*.txt',0**

**db 'All formats',0,'\*.txt;\*.asm;\*.inc;\*.ini',0**

**db 'All files',0,'\*.\*',0,0**

**fn\_saved db 0 ;флаг имени файла 1=сохранено, 0=нет**

**errtxt db 'Код ошибки: %u',0**

**errbuf rb $-errtxt+10**

**fname rb MAXSIZE**

**hfile dd ?**

**hheap dd ?**

**pmem dd ?**

**sbuf dd ?**

**hwnd dd ?**

**hmenu dd ?**

**hedit dd ?**

**hacc dd ?**

**font dd ?**

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

**ofn OPENFILENAME sizeof.OPENFILENAME,0,0,filter,0,0,0,fname,MAXSIZE**

**msg MSG**

**client RECT**

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

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

**start:**

**invoke GetModuleHandle,0**

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

**mov [ofn.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\_CLOSE**

**je .EXIT**

**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:**

**call get\_modified**

**invoke SendMessage,[hedit],WM\_SETTEXT,0,0**

**mov [fn\_saved],0**

**jmp .finish**

**.OPEN:**

**call get\_modified**

**call open\_file**

**jmp .finish**

**.SAVE:**

**call save\_file**

**jmp .finish**

**.SAVEAS:**

**call get\_save**

**jmp .finish**

**.EXIT:**

**call get\_modified**

**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**

**.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 SendMessage,[hedit],EM\_LIMITTEXT,MEMSIZE-1,0**

**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**

**get\_modified:**

**invoke SendMessage,[hedit],EM\_GETMODIFY,0,0**

**cmp eax,0**

**je .not\_modified**

**invoke MessageBox,[hwnd],saveq,title,MB\_YESNO + MB\_ICONWARNING**

**cmp eax,IDYES**

**jne .not\_modified**

**call save\_file**

**.not\_modified:**

**retn**

**save\_file:**

**cmp [fn\_saved],1**

**je create\_file**

**get\_save:**

**mov [ofn.Flags],OFN\_EXPLORER + OFN\_OVERWRITEPROMPT**

**invoke GetSaveFileName,ofn**

**cmp eax,0**

**je failed**

**create\_file:**

**invoke CreateFile,fname,GENERIC\_READ + GENERIC\_WRITE,FILE\_SHARE\_READ + FILE\_SHARE\_WRITE,0,CREATE\_ALWAYS,FILE\_ATTRIBUTE\_NORMAL,0**

**mov [hfile],eax**

**invoke GetProcessHeap**

**mov [hheap],eax**

**invoke HeapAlloc,[hheap],HEAP\_ZERO\_MEMORY,MEMSIZE**

**mov [pmem],eax**

**invoke SendMessage,[hedit],WM\_GETTEXT,MEMSIZE,[pmem]**

**invoke lstrlen,[pmem]**

**invoke WriteFile,[hfile],[pmem],eax,sbuf,0**

**mov [fn\_saved],1**

**invoke SendMessage,[hedit],EM\_SETMODIFY,0,0**

**invoke HeapFree,[hheap],0,[pmem]**

**invoke CloseHandle,[hfile]**

**retn**

**open\_file:**

**mov [ofn.Flags], OFN\_FILEMUSTEXIST + OFN\_PATHMUSTEXIST + OFN\_EXPLORER**

**invoke GetOpenFileName,ofn**

**cmp eax,0**

**je failed**

**invoke CreateFile,fname,GENERIC\_READ + GENERIC\_WRITE,FILE\_SHARE\_READ + FILE\_SHARE\_WRITE,0,OPEN\_EXISTING,FILE\_ATTRIBUTE\_NORMAL,0**

**mov [hfile],eax**

**invoke GetProcessHeap**

**mov [hheap],eax**

**invoke HeapAlloc,[hheap],HEAP\_ZERO\_MEMORY,MEMSIZE**

**mov [pmem],eax**

**invoke ReadFile,[hfile],[pmem],MEMSIZE-1,sbuf,0**

**invoke SendMessage,[hedit],WM\_SETTEXT,0,[pmem]**

**mov [fn\_saved],1**

**invoke HeapFree,[hheap],0,[pmem]**

**invoke CloseHandle,[hfile]**

**retn**

**failed:**

**retn**

**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',\**

**comdlg32,'COMDLG32.DLL'**

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

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

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

**include 'api\comdlg32.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**

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

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

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

**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',<'Простой Текстовый Редактор',0Dh,0Ah,'Vasylenko Roman Andreevich. 2021.v01.'>,-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 компилятора 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','0.2.0.0',\**

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

**; EOF**

В самом начале объявляются две константы: MAXSIZE и MEMSIZE. Первая — это максимальное количество байт под полный путь и имя файла, вторая — размер памяти, выделяемой под буфер для текста, включая завершающий строку ноль. В данной ситуации максимальное количество символов в тексте у нас не должно превышать 65 536 (64 Кб). Памяти выделяется на один байт больше, но в окне редактирования и в файле на жестком диске завершающий ноль записан не будет. При необходимости вы всегда можете изменить данную константу по своему усмотрению в пределах от 2 до 2 147 483 647 (7FFFFFFFh). Потому что максимальное количество символов, которое может вместить стандартный элемент EDIT = 7FFFFFFEh, а минимальное = 1. В секции данных мало нового. Фильтр допустимых файлов (filter) состоит из одной или нескольких пар строк, завершающихся нулем. Каждая пара — это отображаемое имя фильтра и собственно сам фильтр. Последняя пара должна завершаться двумя нулями. Переменную fnsaved мы будем использовать для того, чтобы определять, сохранено ли текущее имя файла в переменной fname. Еще у нас появилась структура OPENFILENAME. Список ее элементов вы можете увидеть в \INCLUDE\EQUATES\ COMDLG32.INC. Чтобы не париться с нудным описанием каждого элемента, опишу только те, которые могут нам понадобиться в ближайшем будущем, тем более что последние 11 элементов я даже не инициализировал какими-либо значениями (последние элементы структуры обычно автоматически инициализируются нулями, если опущены). Хотя по правилам хорошего тона необходимо было бы дописать после MAXSIZE еще 11 ноликов через запятую, но это уж кому как нравится.

lStructSize — размер структуры в байтах; hwndOwner — дескриптор окна-владельца диалогового окна открытия файла; hInstance — дескриптор исполняемого модуля; lpstrFilter — указатель на фильтр; lpstrCustomFilter — указатель на пользовательский фильтр; nMaxCustFilter — размер пользовательского фильтра; nFilterIndex — номер фильтра, выбираемого по умолчанию; lpstrFile — указатель на полное имя файла; nMaxFile — размер буфера для полного имени файла; lpstrFileTitle — короткое имя файла (только имя, без пути к файлу); nMaxFileTitle — размер буфера для короткого имени; lpstrInitialDir — открываемая по умолчанию директория; lpstrTitle — заголовок диалогового окна открытия/сохранения файла.  
  
Переходим к секции кода. Начало похоже на предыдущую версию, только надо не забыть скопировать дескриптор исполняемого модуля еще и в ofn.hInstance. Затем добавляем обработку сообщения WM\_CLOSE, чтобы по команде закрытия можно было проверить, изменилось ли содержимое окна, и предложить сохранение, если изменилось.  
  
Обработчики сообщений от элементов меню "Файл" теперь доступны (параметры MFS\_GRAYED сняты с этих элементов в ресурсе menu) и готовы обрабатывать свои сообщения. При выборе пункта "Создать" (IDM\_NEW -> .NEW), происходит вызов подпрограммы (метка get\_modified). Команда call (от англ. call — вызывать) очень похожа на команду jmp. Только команда call сохраняет при этом в стек адрес возврата — адрес следующей за call команды, чтобы потом команда ret (от англ. return — возвращаться) могла вернуть управление на команду, следующую за call. Пара команд call — ret используется для возможности вызова какой-либо подпрограммы из разных мест программы. Например, макрос invoke, к которому мы уже так привыкли, тоже использует команду call, только предварительно запихивает в стек параметры вызываемой функции. На "том конце провода" функция API выполняет необходимые операции и при помощи команды ret возвращает управление следующей команде нашей программы. Сложно? Ерунда! Это я вам еще только в общих чертах обрисовал картину. Ну да ладно, не стоит на этом загоняться — это уже почти хакерский уровень, а вы пока что как бы "чайники". Запомните пока просто, что call вызывает, а ret возвращает.  
  
Вот и в нашей ситуации выполнение команд временно перескакивает на метку get\_modified. Там посылкой сообщения EM\_GETMODIFY нашему окну редактирования мы узнаем, был ли изменен текст окна: если изменен, вернется единица, иначе — ноль. Параметры у сообщения отсутствуют, а потому выставляются в ноли. Если изменений нет, то значит и сохранять нечего — возвращаемся. Retn (Return Near) — это тот же ret, только для близких возвратов — мы же находимся внутри процедуры WindowProc. Если же изменения имели место, то переспрашиваем у пользователя, следует ли их сохранять. Если следует (IDYES), то вызываем save\_file, нет — возвращаемся. На метке save\_file проверяется, сохранен ли путь к нашему файлу (мы сами будем устанавливать переменную fnsaved в единицу, когда путь сохранен или файл открыт, и в ноль при создании нового файла, путь которого еще не определен). Если путь сохранен, сразу переходим к метке create\_file, если нет — вызываем диалог сохранения файла функцией  
GetSaveFileName. Эта функция имеет лишь один параметр (но зато какой!) — указатель на огромную структуру OPENFILENAME. Предварительно устанавливаем флаги OFN\_EXPLORER для открытия диалога в стиле проводника windows и OFN\_OVERWRITEPROMPT для подтверждения перезаписи файла, если он существует. GetSaveFileName возвращает ноль в случае, если пользователь отменил сохранение. Иначе в буферы помещается имя файла, а возвращаемое значение отлично от ноля. Таким образом, на метке create\_file, в fname содержится полное имя файла.  
  
Функция CreateFile создает или открывает объект ввода/вывода. Параметры:  
1. Имя объекта.  
2. Права доступа: GENERIC\_READ — чтение, GENERIC\_WRITE — запись или оба сразу.  
3. Права на совместный одновременный доступ к файлу несколькими процессами: 0 — доступ другим процессам запрещен до закрытия файла, FILE\_SHARE\_READ — разрешено чтение, FILE\_SHARE\_WRITE — разрешена запись, FILE\_SHARE\_DELETE — разрешено удаление.  
4. Атрибуты безопасности: 0 для значений по умолчанию.  
5. Способ открытия файла: CREATE\_NEW — создать новый файл, ошибка, если файл существует, CREATE\_ALWAYS — создать новый файл, перезаписывает старый, если файл существует, OPEN\_EXISTING — открыть файл, ошибка, если файл не существует, OPEN\_ALWAYS — открыть файл, если файл не существует, он будет создан, TRUNCATE\_EXISTING — открыть файл и очистить его содержимое, ошибка, если файл не существует.  
6. Набор атрибутов (скрытый, системный и т.д.).  
7. Файл-шаблон атрибутов.  
  
При успешном создании/открытии файла возвращается его дескриптор, если файл отсутствует — возвращается 0, в случае ошибки возвращается -1. Функция GetProcessHeap возвращает дескриптор кучи вызывающего функцию процесса. Куча — это область виртуальной памяти, под которую не выделяется реальная физическая память, но по мере заполнения кучи данными диспетчер, управляющий кучами (heap manager), выделяет под нее физическую память. Функция HeapAlloc выделяет блок памяти в куче и возвращает указатель на выделенный блок. Параметры: дескриптор кучи; флаги способа выделения памяти (HEAP\_ZERO\_MEMORY — проинициализировать блок нулевыми значениями); размер выделяемого блока в байтах. После выделения блока мы копируем текст, завершающийся нолем, из окна редактирования в этот блок, отсылкой окну редактирования сообщения WM\_GETTEXT. Функция lstrlen (параметр — адрес строки) возвращает длину строки в символах без учета завершающего ноля. Функция WriteFile записывает данные в файл. Ее параметры: дескриптор файла; указатель на данные; размер данных в байтах; буфер для ответа, сколько байт удалось записать; указатель на структуру OVERLAPPED для дополнительных сведений. В случае ошибки возвращается ноль. Записываем в fnsaved единицу — файл сохранен и его имя хранится в переменной fname. Отсылаем окну редактирования сообщение EM\_SETMODIFY — первый параметр 0 означает, что содержимое окна не изменялось, второй параметр не используется. Функция HeapFree освобождает указанный блок памяти в куче. Параметры: дескриптор кучи; флаги; указатель на блок памяти. Функция CloseHandle закрывает объект (в нашем случае — открытый файл), чей дескриптор указан в единственном параметре. Теперь возвращаемся к следующей после call команде.  
  
При создании файла после вызова подпрограммы get\_modified мы очищаем содержимое окна редактирования и устанавливаем fnsaved в ноль, так как имя нового файла пока неизвестно. При открытии файла мы опять же вызываем подпрограмму сохранения старого файла (get\_modified), а затем вызываем подпрограмму открытия (open\_file). Там все почти аналогично подпрограмме get\_modified. Устанавливаем флаги OFN\_FILEMUSTEXIST (открываемый файл должен существовать), OFN\_PATHMUSTEXIST (открываемый путь к файлу должен существовать) и OFN\_EXPLORER. Функция GetOpenFileName вызывает диалог открытия файла и возвращает его имя в буфер fname. Открываем выбранный файл функцией CreateFile. Выделяем память. Функция ReadFile аналогична функции WriteFile, только не пишет данные в файл, а читает их. Размер данных для чтения указываем MEMSIZE-1, потому что последний байт в памяти необходимо оставить нулевым — это завершающий строку ноль. Устанавливаем прочитанные данные текстом окна редактирования (WM\_SETTEXT). Задвигаем в fnsaved единицу — файл открыт и его имя нам известно. Освобождаем память. Закрываем файл.  
  
При выборе пункта "Сохранить" подпрограмма сохранения вызывается с метки save\_file, дабы избежать вопроса о необходимости сохранения. При выборе пункта "Сохранить как…" вызываем сохранение с метки выбора файла для сохранения.  
  
Выбор пункта "Выход" или закрытие программы приведет к выполнению get\_modified и вызову функции **DestroyWindow** с дескриптором окна в качестве единственного параметра.  
На метке failed ничего не происходит — возврат без выполнения каких-либо действий. Здесь вы самостоятельно можете добавить дополнительную обработку ошибок и вывод соответствующего сообщения. Также можете на свое усмотрение внести проверку успешного завершения после каждой важной функции подпрограмм чтения и записи файла.  
Согласен, что некоторые моменты разъяснены не так подробно, как хотелось бы, но тут я уповаю на вашу сообразительность. Если что-то непонятно — возможно стоит еще раз прочесть предыдущие уроки. В любом случае не падайте духом и не теряйте уверенности в своих силах и возможностях.  
Все приводимые примеры были протестированы на правильность работы под Windows XP и скорее всего будут работать и под другими версиями Windows, однако я не могу дать никаких гарантий их правильной работы на вашем компьютере.