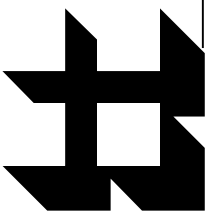

ГЛАВА 28

Полезные советы

При создании собственного проекта возникает так много вопросов! В данной главе собраны ответы на некоторые из них. Мы надеемся, эта коллекция со временем будет пополняться.

Самый главный совет: Даже если у вас что-то не получается сразу, никогда не отчаивайтесь! Всегда помните о том, что Visual FoxPro, как правило, предлагает несколько вариантов решения проблемы. Поэтому просто попытайтесь посмотреть на проблему с другой стороны, применить другой метод.

В крайнем (да и не в крайнем!) случае можно призвать на помощь коллективный разум, обратиться за помощью на один из многочисленных форумов программистов в Интернете, например на **www.FoxClub.ru**, который славится доброжелательным отношением к новичкам.

1. Как изменить кодовую страницу таблицы

Начнем с ситуации, которая встречается достаточно часто: вы создали в VFP 9.0 какую-то таблицу, наполнили ее данными, потратили на это приличное время, и вдруг (не удивляйтесь, это, как правило, происходит именно "вдруг") выясняется, что заказчикам нужна эта же таблица, но в FoxPro для DOS, например, для Fox 2.6. Как известно, Fox 2.6 имеет кодовую страницу по умолчанию 866, а Visual FoxPro 9.0 — 1251. Заказчик не сможет открыть вашу таблицу, на экране появится сообщение "Not a database file".

Что же делать?

Нужно вернуться в VFP 9.0 и выполнить в командном окне следующие команды:

```
codepage=866
use a2
copy to c:\temp\a3 fox2x as 866
```

Теперь созданную таблицу можно открыть и в DOS FOX.

Изменить кодовую страницу с 866 на 1251 можно так:

```
use way
? cpdbf()
copy to way_win as 1251
use way_win
? cpdbf()
```

Функция `cpdbf()` возвращает кодовую страницу для открытой таблицы.

Если же вам просто нужно сменить кодовую страницу у таблицы, — в 30 байте любым редактором, например, hiew изменить `C9` на `65` (меняется с 1251 на 866).

> ***Примечание***
>
> Кодовую страницу можно указывать и для переменных:
>
> `CPCONVERT(nCurrentCodePage, nNewCodePage, cExpression)`

Вместе с VFP поставляется огромное число классов, примеров решений, готовых форм. Поэтому многим программистам приходит разумное желание их использовать, заменив таблицы на нужные. В некоторых случаях наблюдается следующий эффект. Содержание таблиц просматривается нормально. Однако при изменениях на форме и сохранении содержания часть текста портится. Некоторые русские буквы отображаются неверно. Причина проста. Формы или библиотеки классов для этих форм *также могут быть открыты как таблицы*. Поскольку VFP поставляется в кодировке 1252, то и формы, и классы также имеют эту же кодировку. Исправить форму можно двумя способами. Открыв как таблицу и переписав в кодировке 1251, например,

```
use form1.scx
copy to c:\temp\form1.scx as 1251
use in form1
```

Указывать расширение файла при этом обязательно, поскольку по умолчанию VFP полагает, что расширение — dbf.

Другим способом смены кодовой страницы таблиц, форм, библиотек классов является поставляемая вместе с языком утилита tool\cpzero\CPZERO.PRG. Ее достаточно запустить с именем файла, в котором требуется изменить кодовую страницу. Можно указать и желаемую кодовую страницу.

2. Как прочитать список файлов, расположенных в одном каталоге

Самый простой способ — использовать функцию `ADIR()`(листинг 28.1).

```
Set Defa to [c:\Program Files]
qf=adir(adf,"C:\Program Files\"+"*.*","D")
* массив adf будет содержать информацию обо всех файлах указанного каталога
(с подкаталогами, поскольку применена опция "D").
```

```
** adf[1,1] — имя файла
** adf[1,2] — размер
** adf[1,3] — дата создания
** adf[1,4] — время создания
** adf[1,5] — атрибуты файла
=asort(adf,2,-1,1)
ab=0
for j=1 to qf
   ab=ab+adf[j,2]  && подсчитывает общий размер всех файлов
next
=ASORT(adf,2) && сортируем данные (имя массива, начальный элемент, -1 — сортировать
все элементы, тип сортировки — Ascending, т. е. по возрастанию)
CREATE TABLE table1 (Name c(50))  && создаем пустую таблицу для имен файлов
FOR j=1 To qf   && в этом цикле данные читаются из массива в таблицу
        m.name=adf[j,1]
        INSERT INTO table1 FROM Memv
NEXT
* а теперь передаем данные из таблицы в текстовый файл
SELECT * FROM table1 ORDER BY name TO FILE a1.txt
USE         && закрываем таблицу
MODIFY FILE a1.txt NOEDIT  && смотрим на то, что получилось
CANCEL         && вот и все, собственно
```

3. Вы хотите закрыть свой проект от взлома, хотя бы от не очень компетентных людей

Прежде всего, заметим, что закрывать проект — идея не из лучших. Если ваш проект представляет интерес для хакеров, его все равно взломают, разберут побайтно и побитно. Интересно, что даже такие, казалось бы, эффективные средства, как Guardant Wizard (программа, входящая в комплект с электронными ключами Guardant, некогда Novex), не дают абсолютно никакой защиты от декомпиляции.

Декомпилируется все — формы, меню, FXP-файлы и пр. Всякая защита как ставится, так и снимается.

Но в некоторых случаях такие идеи имеют право на жизнь.

Наиболее популярным декомпилятором программ на VFP является ReFox.

Защититься от ReFox можно самим ReFox. В самом начале головного модуля программы, после параметров (если они указаны), нужно написать следующий код:

```
IF .F.
   _refox_ = (9876543210)
ENDI
```

В опциях ReFox ставим `Branding Level II`, далее F6 (Brand) на *.exe, который надо закрыть от декомпиляции, затем вводим пароль, и все! Конечно, велика вероятность, что хакеры его вмиг вскроют, но зато это очень простой и действенный способ для защиты своей работы от не очень компетентных людей. Одновременно это является неплохим способом восстановить исходные тексты своих программ, если вы их потеряли.

Главное — пароль не забыть!

Еще один популярный способ подходит для случая, когда приложение работает только на одном компьютере. В этом случае есть возможность "привязать" проект к этому компьютеру: к серийному номеру диска, к его материнской плате или другим составным частям.

4. Вы хотите ввести пароль и зашифровать его

Скорее всего, вы хотите получить следующую картинку (рис. 28.1).

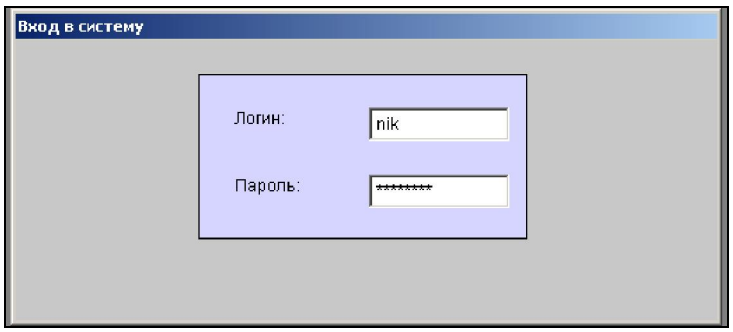

Рис. 28.1. Диалоговое окно ввода пароля

Для того чтобы вводимые значения заменялись звездочками, достаточно установить в `Property` вводимого `TextBox` параметр `PasswordChar = *`.

Вариантов применения какого-либо шифра, чтобы введенный пароль (и не только пароль!) хранился в таблице в зашифрованном виде, бесчисленное множество. Существуют шифры типа CRC16, реализуемые функцией SYS(2007), CRC32 (также через SYS(2007)), MD5 и прочие из семейства MD-хэшей — (доступно через вызовы функций CryptoAPI, для чего реализован FFC-класс, или через специальные Cripto-DLL). Казалось бы, изобретать свое шифрование/хэширование смешно. Однако мы предлагаем несложный пример шифрования и дешифрования, который применим для простых случаев, когда в серьезной защите нет необходимости, а хранить в открытом виде пароли для входа в программу нежелательно, поскольку любую таблицу легко просмотреть (листинги 28.3 и 28.4).

Итак, у нас должны быть два `TextBox`: один для ввода идентификатора пользователя, второй для ввода пароля. В событии `KeyPress TextBox`, предназначенного для ввода пароля, поместим следующий код (листинг 28.2).

```
LPARAMETERS nKeyCode, nShiftAltCtrl
IF LASTKEY()=13
   SET PROCEDURE TO mybrow
```

```
Local m.var
   m.var=ALLTRIM(thisform.edit2.value) && введенный пароль
   m.var1=ALLTRIM(thisform.edit1.value) && введенный логин

   SELECT users       && таблица для хранения пользователей и их паролей
   IF SEEK(m.var1,"users")
IF psw_code(m.var)<>ALLTRIM(users.pass)
   WAIT WINDOW ('Неверный пароль!') TIMEOUT 3
      quit
ELSE
   WAIT WINDOW users.fio time 3    && показать на экране логин
   thisform.Release()
ENDIF
ELSE
   WAIT WINDOW ('Неизвестный пользователь!Обратитесь к администратору.') TIMEOUT 3
      quit
endif
thisform.edit2.Value=""
thisform.edit2.SetFocus()
RELEASE m.var
endi
```

В процедуре `My_Dec` находится само шифрование (и дешифрование тоже).

```
PROCEDURE My_Dec
FUNC psw_code
 PARA pswtext
 pswtext = UPPER(pswtext)
 PRIV i
 r_psw = STR(LEN(ALLTRIM(pswtext)), 1)+ALLTRIM(pswtext)
 tr_psw = ''
 FOR i = 1 TO LEN(r_psw)
      code_sym = ASC(SUBSTR(r_psw, i, 1))+10+LEN(r_psw)
      code_sym = IIF(code_sym>255, code_sym-223, code_sym)
      tr_psw = tr_psw+CHR(code_sym)
 ENDF
 m.pass1=tr_psw
 RETU tr_psw
*
FUNC psw_dec
 PARA pswtext
 PRIV i
 tr_psw = ''
 FOR i = 2 TO LEN(pswtext)
      code_sym = ASC(SUBSTR(pswtext, i, 1))-10-LEN(pswtext)
      code_sym = IIF(code_sym<33, code_sym+223, code_sym)
      tr_psw = tr_psw+CHR(code_sym)
 ENDF
 RETU tr_psw
*
```

```
В каталоге FFC Visual FoxPro  есть класс-оболочка над CryptoAPI — _crypt.vcx

*Шифрование
o_cryptapi1=NEWOBJECT("_cryptapi","c:\Program Files\Microsoft Visual FoxPro
9\FFC\_crypt.vcx") && здесь укажите путь на FFC

lcEncryptedString = ""
lcPassword="MyPassword"
lcDecryptedString="Hello!"
o_cryptapi1.encryptsessionstreamstring(lcDecryptedString, lcPassword,
@lcEncryptedString)
? lcEncryptedString
o_cryptapi1=null

*Дешифрование
o_cryptapi1=NEWOBJECT("_cryptapi","d:\vfp\common\libs\_crypt.vcx")
o_cryptapi1.decryptsessionstreamstring(lcEncryptedString, lcPassword,
@lcDecryptedString)
? lcDecryptedString
o_cryptapi1=null
```

5. Вы хотите, чтобы ваше приложение запускалось только один раз на каждом компьютере

Многие программисты сетуют на пользователей их программ, что они некомпетентны, запускают на своем компьютере несколько экземпляров приложения сразу, часто это вызывает ошибки работы приложений. Для ликвидации такой ситуации есть несколько решений. Так, в головном модуле программы можно поместить следующий текст (листинг 28.5).

```
If ("EXE" $ SYS(16,0)) .AND. is_run32('Delo')
        Is_Task = .T.
        Messagebox('Аналогичное приложение уже запущено с этого
компьютера','')
                close all
                clear all
                Return
Endif

Текст программы is_run32 имеет следующий вид:
Function is_run32

Lparameter pctitle

**** запущена ли форма
Declare INTEGER GetActiveWindow IN Win32API
Declare INTEGER GetWindow IN Win32API;
        INTEGER hwnd, INTEGER dflag
```

```
Declare INTEGER GetWindowText IN Win32API ;
        INTEGER hwnd, STRING @lptstr, INTEGER cbmax
Declare INTEGER CloseWindow IN Win32API;
        INTEGER hwnd
Declare INTEGER DestroyWindow IN Win32API;
        INTEGER hwnd
Local lnhwnd, lnnext, lldone, lctitle_bar, lcsearchfor,;
        lntext_len
lcsearchfor = UPPER(ALLTRIM(pctitle))
lnhwnd = GetActiveWindow()
lnnext = 2
lctitle_bar = ""
Do WHILE lnhwnd # 0
        If type('lnhwnd') # 'N'
                Loop
        Endif

        lctitle_bar = SPACE(200) + CHR(0)
        lntext_len = GetWindowText(lnhwnd, @lctitle_bar, 200)
        lctitle_bar = UPPER(LEFT(lctitle_bar, lntext_len))
        lldone = (lcsearchfor $ lctitle_bar)
        lp=lnhwnd
        lnhwnd = iif(lldone, 0, GetWindow(lnhwnd, lnnext))
Enddo
Return lldone
```

Некоторые программисты помещают в базу данных специальную таблицу, содержащую имя компьютера и пользователя.

При входе в программу проверим наличие в этой таблице записи со значением sys(0). Если значение найдено, программа останавливается, если не найдено, пишем в эту таблицу значение этой функции и продолжаем работу. При выходе из программы стираем эту запись. Таким образом, один пользователь не может запустить более одного экземпляра программы. Проблемой такого решения является аварийный выход из программы, скажем, по зависанию компьютера, когда запись вошедшего пользователя не удалена.

6. Как вычислить определенные даты

```
*  Первый день месяца
ldStart = (Date()+1) — DAY(Date())
*  Последний день месяца
lcEnd = GOMONTH((Date()+1)-DAY(Date()),1)-1
```

или

```
lcEndx = GOMONTH(ldstart,1)-1
```

7. Как сделать всплывающую подсказку

Чтобы у вас появлялась всплывающая подсказка, нужно:

1. В свойствах самой формы определить `SHOWTIPS=.T.`

   На нужном объекте должен находиться фокус, т. е. не помешает строчка `Tisform.Grid1.SetFocus()`.

2. В свойстве `Grid ToolTipText` необходимо указать сам всплывающий текст: Thisform.Grid1.Tooltiptext="Этот текст нам необходим".

8. Как работать с INI-файлами

INI-файл или файл с расширением ini (от слов *initialization file* — файл инициализации) — это обычный текстовый файл, который имеет специфическое содержание.

Выглядит его содержимое примерно так:

```
// Комментарий
[Имя_раздела_1]
Имя_реквизита_1 = значение реквизита
Имя_реквизита_2 = значение реквизита

// Комментарий
[Имя_раздела_2]
Имя_реквизита_1 = значение реквизита
Имя_реквизита_2 = значение реквизита
```

В квадратных скобках указывается имя раздела (другое название — "секция"), затем идет список реквизитов (другое название — "ключ") этого раздела и их значение. Количество разделов и количество реквизитов внутри раздела не ограничено. Желательно давать имена разделам и реквизитам в одно слово из латинских букв без пробелов и специальных символов, но это скорее перестраховка. Если для вас удобнее воспринимать имя раздела или реквизита с пробелом (в два слова) — пишите с пробелом.

В качестве значения реквизита принимается вся строка от первого отличного от пробела символа после символа равенства до окончания строки, включая пробелы, т. е. здесь для задания текстовой строки использовать кавычки не надо. Предполагается, что все значения имеют только и исключительно символьный тип.

Символом комментария обычно служат две наклонные черты подряд. Хотя, в принципе, можно использовать абсолютно любой символ (вот только пробел нельзя). Для используемых ниже API-функций важен факт совпадения искомого имени, начиная с первого символа. Достаточно того, чтобы первый символ не совпадал с искомым.

В принципе можно прочитать такой файл из FoxPro функциями чтения текстовых файлов (`FOPEN()`, `FREAD()`, `FileToStr()` и т. п.) и сделать его разбор, но есть способ лучше. Существуют специальные API-функции, предназначенные для чтения и записи INI-файлов.

- `GetPrivateProfileString` — считывает значение указанного реквизита из указанного раздела.

- `WritePrivateProfileString` — записывает значение указанного реквизита в указанный раздел указанного INI-файла. Если указанного раздела или реквизита не существует, то он будет создан. Если указанного INI-файла не существует, то он также будет создан.

Выглядит это примерно так (листинги 28.6 и 28.7).

```
Declare Integer WritePrivateProfileString In Win32API as WritePrivStr;
String cSection, ;  && имя раздела
String cKey, ; && имя реквизита
String cValue, ; && значение реквизита
String posfile && имя INI-файла с полным путем доступа
Local lnError
lnError= WritePrivStr("TestSection", ;
                      "TestKey", ;
                      "TestValue", ;
                      Fullpath('')+"test.ini")
IF lnError<>1
        * Запись в INI-файл не удалась. Произошла ошибка
  ELSE
        * Смотрим, что получилось во вновь созданном файле
        MODIFY FILE test.ini
ENDIF
```

```
Declare Integer GetPrivateProfileString In Win32API  As GetPrivStr ;
        String cSection, ;               && Имя раздела
        String cKey, ;           && Имя реквизита
String cDefault, ;       && Значение по умолчанию, если нет указанного раздела или
реквизита
        String @cBuffer, ;       && Собственно считанное значение реквизита
Integer nBufferSize, ;   &&Максимальное количество символов в считанном реквизите
        String posfile           && имя INI-файла с полным путем доступа

  LOCAL lcBuffer, lnBuffer
  lcBuffer = SPACE(2000)
  lnBuffer = GetPrivStr("TestSection", ;
                        "TestKey", ;
                        "Нет значения", ;
                        @lcBuffer, ;
                        LEN(m.lcBuffer), ;
                        Fullpath('')+"test.ini")

  IF m.lnBuffer = 0
        * Ничего не прочитали
  ELSE
        * Прочитанное значение хранится в первых m.lnBuffer символах переменной
m.lcBuffer
        ?LEFT(m.lcBuffer,m.lnBuffer)
        * Общая длина переменной m.lcBuffer по-прежнему 2000 символов,
```

```
        * но символ m.lnBuffer+1 имеет ASCII-код равный 0, а все прочие — пробелы
  ENDIF
```

Удаление раздела или реквизита

Удаление — это запись неопределенного значения (листинг 28.8).

```
* Удаление только одного реквизита TestKey в разделе TestSection
  LOCAL lnError
  lnError = WritePrivStr("TestSection", ;
                         "TestKey", ;
                         NULL, ;
                         Fullpath('')+"test.ini")
  IF lnError<>1
        * Запись в INI-файл не удалась. Произошла ошибка
  ELSE
        * Смотрим, что получилось
        MODIFY FILE test.ini
  ENDIF

 * Удаление всего раздела TestSection со всеми реквизитами
  LOCAL lnError
  lnError = WritePrivStr("TestSection", ;
                         NULL, ;
                         NULL, ;
                         Fullpath('')+"test.ini")
  IF lnError<>1
        * Запись в INI-файл не удалась. Произошла ошибка
  ELSE
        * Смотрим, что получилось
        MODIFY FILE test.ini
  ENDIF
```

Определение структуры INI-файла

Если структура INI-файла (т. е. имена секций и ключей) заранее не известна, то получить структуру INI-файла можно, используя функцию `GetPrivateProfileString`, указав вместо имени секции или ключа число 0. В этом случае в качестве возвращаемого значения вы получите либо список имен секций (если указать 0 первым параметром), либо список имен ключей указанной секции (если указать 0 вторым параметром). Имена будут разделены символом `CHR(0)` (листинг 28.9).

```
* Список секций INI-файла
  LOCAL lcBuffer, lnBuffer
  lcBuffer = SPACE(2000)

  lnBuffer = GetPrivStr(0, ;
                         0, ;
                         "", ;
```

```
                        @lcBuffer, ;
                        LEN(m.lcBuffer), ;
                        Fullpath('')+"test.ini")

  IF m.lnBuffer > 0
        m.lcBuffer = CHR(0) + LEFT(m.lcBuffer,m.lnBuffer)
        LOCAL lnI, lnFromPos, lnToPos
        FOR m.lnI = 1 TO OCCURS(CHR(0),m.lcBuffer)-1
                lnFromPos = AT(CHR(0),m.lcBuffer,m.lnI)
                lnToPos = AT(CHR(0),m.lcBuffer,m.lnI+1)
                ?SubStr(m.lcBuffer,m.lnFromPos+1,m.lnToPos-m.lnFromPos-1)
        ENDFOR
  ENDIF
* Аналогично получаем список ключей секции TestSection

  LOCAL lcBuffer, lnBuffer
  lcBuffer = SPACE(2000)

  lnBuffer = GetPrivStr("TestSection", ;
                        0, ;
                        "", ;
                        @lcBuffer, ;
                        LEN(m.lcBuffer), ;
                        Fullpath('')+"test.ini")

  IF m.lnBuffer > 0
        m.lcBuffer = CHR(0) + LEFT(m.lcBuffer,m.lnBuffer)
        LOCAL lnI, lnFromPos, lnToPos
        FOR m.lnI = 1 TO OCCURS(CHR(0),m.lcBuffer)-1
                lnFromPos = AT(CHR(0),m.lcBuffer,m.lnI)
                lnToPos = AT(CHR(0),m.lcBuffer,m.lnI+1)
                ?SubStr(m.lcBuffer,m.lnFromPos+1,m.lnToPos-m.lnFromPos-1)
        ENDFOR
  ENDIF
```

Разумеется, объявлять API-функции каждый раз перед записью или чтением значений нет необходимости. Достаточно объявить эти функции один раз в стартовом файле.

Применительно к FoxPro следует рассмотреть возможность хранения дополнительных настроек в обычном файле DBF, поскольку FoxPro предназначен, прежде всего, именно для работы с файлами DBF.

9. Как сделать ссылку

В событии `MouseEnter` пишем следующий код (листинг 28.10).

```
LPARAMETERS nButton, nShift, nXCoord, nYCoord
This.ForeColor=Rgb(0,0,198)
This.FontUnderline=.T.
```

```
В событии MouseLeave отменяем установки:

LPARAMETERS nButton, nShift, nXCoord, nYCoord
This.ForeColor=Rgb(0,0,0)
This.FontUnderline=.F.
```

10. Как "прикрепить" созданную вами иконку к проекту

"Прикрепление" иконки к EXE-файлу можно организовать двумя способами:

- Программно, в главном (стартовом) файле дать команду

  ```
  _SCREEN.icon = "MyIcon.ico"
  ```

- Воспользоваться пунктом основного меню **Project** | **Project info** — закладка **Project** — поставить флажок в пункте **Attach icon** и выбрать нужный значок.

Программное задание иконки имеет более высокий приоритет, т. е. этот способ перекроет иконку, выбранную в окне свойств проекта.

Однако каким бы способом вы ни воспользовались, необходимая иконка должна быть доступна к моменту ее использования. Лучше включить ее внутрь проекта.

Ассоциируемая иконка должна обладать следующими свойствами:

- иконка — это именно иконка, т. е. файл с расширением ico;
- иконка (сам файл ico) должна содержать в себе две картинки — 16×16 пикселов и 32×32 пикселов;
- иконка должна иметь столько цветов, сколько поддерживает текущая версия;
- FoxPro, так и операционная система Windows;
- файл EXE должен быть собран (создан) в той версии операционной системы Windows, которая указана в системных требованиях к вашей версии FoxPro.

В принципе FoxPro поддерживает отображение иконок 3 размеров: 16×16, 32×32 и 48×48. Однако размер 48×48 практически нигде не используется. Впрочем, можете также включить иконку размера 48×48 в свой файл ICO. Хотя практического смысла это не имеет.

Также FoxPro поддерживает изображение иконок в 4 наборах цветов: 16 Color, 256 Color, True Color (16 bit), True Color (24 bit). Для своей иконки лучше использовать минимальный цветовой набор — 16 Color или 256 Color. Мало ли, какие установки будут сделаны на машине клиента. А 256 Color поддерживают практически все видеокарты.

Если в системных требованиях для вашей версии FoxPro указано, что разработка приложения должна осуществляться в версии Windows 2000 или Windows XP, то не стоит пытаться собрать готовый файл EXE в младшей версии, например, в Windows 98. Вполне возможно, что вы получите работоспособный файл EXE, но,

скорее всего, "в комплекте" вы получите массу неприятностей, например, не будет отображаться присоединенная иконка.

11. Как организовать *ProgressBar* ("градусник")

Загрузка большого числа файлов или выполнение других длительных операций может быть достаточно продолжительной по времени и поэтому часто сопровождается выводом на экран индикатора `ProgressBar`. Для построения такого "термометра" обычно используют элемент ActiveX Microsoft ProgressBar.

Кроме того, можно использовать в качестве такого "термометра" строку состояния (листинг 28.11).

```
cTextFileName="C:\1.txt" && необходимо изменить на свой файл
LOCAL cPath
LOCAL cFIO
LOCAL cADR
LOCAL cPriz
nFHandl = FOPEN(cTextFileName,0)
  cSetBar = SET("Status Bar")
  SET STATUS BAR ON
  SET STATUS ON
  LOCAL nStrCount,nReadStr
  nStrCount = 10000 && предположим, количество записей
                         && указано в начале текстового файла
  nReadStr = 0
  _SCREEN.ADDOBJECT("sp","Shape")
  _SCREEN.sp.TOP = _SCREEN.HEIGHT-_SCREEN.sp.HEIGHT
  _SCREEN.sp.WIDTH = 0
  _SCREEN.sp.BACKCOLOR = RGB(0,0,255)
  _SCREEN.sp.VISIBLE = .T.
  DO WHILE !FEOF(nFHandl)

        cTmpString = FGETS(nFHandl)
        nReadStr = nReadStr + 1
        DO CASE
        CASE cPATH $ cTmpString
        CASE cFIO $ cTmpString
        CASE cAdr $ cTmpString
        CASE cPriz $ cTmpString
        ENDCASE
        _SCREEN.sp.WIDTH = 4*INT(nReadStr*100/nStrCount)
        _SCREEN.sp.REFRESH
  ENDDO
  =FCLOSE(nFHandl)
  WAIT "" TIMEOUT 0.1
  SET MESSAGE TO
  _SCREEN.REMOVEOBJECT("sp")
  IF cSetBar = "OFF"
        SET STATUS BAR OFF
```

```
  ENDIF
```

12. Обновление версий

Довольно трудно своевременно уследить за распространением каждой новой версии приложения в большой сети, особенно если приложение только что написано, и версии следуют одна за другой. В этом случае можно применить автоинкрементирующуюся нумерацию версий и осуществлять проверку, требуется ли установка новой версии на конкретной машине. На сервере создайте таблицу с номером версии. Обычно он записывается в формате 00.00.0000, например, 01.09.7990. На компьютере пользователя разместите INI-файл с текущим номером версии. При запуске программы проверяйте

```
If      UpgradeVersion > (App.Major, "00")+ "."+(App.Minor, "00")+ "."+;
        (App.Revision, "0000")
        Do upgrade              && запуск апгрейда с сетевого диска
EndIf
```

Если версия в INI-файле ниже, чем версия, записанная в таблице на сервере, то автоматически запустится программа установки новой версии.

13. Преобразовать "Пупкин Василий Иванович" в "Пупкин" "Василий" "Иванович"

Если имеется поле таблицы с данными, имеющими какой-либо разделитель, можно использовать функцию `Getwordnum`. Параметрами функции являются символьная строка поиска, номер слова в строке, которое требуется выделить, а также разделитель. Разделителем может быть не только пробел, но и любой другой символ (листинг 28.12).

```
Tbl1.fld1="Пупкин Василий Иванович"
LnName1=GetWordNum(tbl1.fld1,1," ")
lnName2=GetWordNum(tbl1.fld1,2," ")
lnName3=GetWordNum(tbl1.fld1,3," ")
```

14. Анимированная иконка

Visual FoxPro 9.0 поддерживает отображение анимированных иконок, для этого можно использовать свойство `Picture` элемента управления `Image`. Особенно удобны анимированные иконки для индикации процесса — если иконка движется, значит, процесс не завис. Для создания анимированных иконок можно использовать Macromedia Flash.

В программе перед каким-либо процессом, требующим времени, добавьте следующий код (листинг 28.13).

```
LOCAL laIconList
DIMENSION laIconList[4]
laIconList[1]="icon01.ico"
laIconList[2]="icon02.ico"
laIconList[3]="icon03.ico"
laIconList[4]="icon04.ico"
```

&& Теперь программно создадим объект-таймер:

```
SET CLASSLIB TO animate
THISFORM.ADDOBJECT("tmrIcon","animicon",@laIconList)
```

&& Активизируйте таймер путем установки интервала:

```
THISFORM.tmrIcon.Interval=500 && пол-секунды
```

Замечание

Стандартные картинки с летающими из папки в папку (или корзину) документами можно найти в Visual Studio, в его состав входит набор этих AVI-файлов. Демонстрировать на форме их можно с помощью ActiveX-компонента Microsoft animation control.

15. Как создать задание на SQL-сервере

Можно воспользоваться SQLDMO (SQL Distributed Management Objects). SQLDMO — это распределенный управляющий объект, предназначенный для решения административных задач в SQL-сервере. Поскольку доступ к SQLDMO возможен через интерфейс COM, разработчики программ на VFP могут обратиться к нему и воспользоваться его свойствами, методами и событиями (листинг 28.14).

```
SQLServer=CREATEOBJECT("SQLDMO.SQLServer")
SQLJob = CREATEOBJECT("SQLDMO.Job")
JobStep = CREATEOBJECT("SQLDMO.JobStep")
JobSchedule = CREATEOBJECT("SQLDMO.JobSchedule")
SQLServer.CONNECT("Server", "sa", "")
JobServer = SQLServer.JobServer
WITH JobStep
  .NAME = "DMOStep1"
  .StepId = 1
  .COMMAND = "Select companyname from  northwind..customers"
ENDWITH
JobSchedule.NAME = "DMOSchedule1"

JobSchedule.Schedule.BeginAlter
WITH JobSchedule.Schedule
  .ActiveStartDate = 20061201
  .ActiveStartTimeOfDay = 120000
ENDWITH
```

```
JobSchedule.Schedule.DoAlter
SQLJob.NAME = "DMOJob1"
SQLJob.StartStepId = 1
JobServer.jobs.ADD(SQLJob)
SQLJob.AddStepToJob(JobStep)
SQLJob.JobSchedules.Add(JobSchedule)
SQLServer.DISCONNECT
```

На компакт-диске программа находится в CHAR28\28_14.

Еще один код поможет вам получить список баз данных и таблиц указанного SQL-сервера (листинг 28.15).

```
clear
Try
        sServer = CREATEOBJECT("SQLDMO.SQLServer")
        sServer.Connect("EODServer", "sa", "")
        coudata=sserver.databases.count
FOR i=1 TO coudata
        sDatabase =  sServer.databases.item(i).name
        ? sdatabase
        couTables=sserver.databases.item(i).tables.count
        FOR j=1 TO couTables
        ? sserver.databases(i).tables(j).name
        endfor
endfor
        sServer.DisConnect()
CATCH TO ex
  MessageBox(ex)
EndTry
```

Аналогично можно создать задание на резервное копирование базы данных на SQL-сервере (листинг 28.16).

Листинг 28.16. Организация резервного копирования базы данных на SQL-сервере средствами SQLDMO

```
Try
sServer = CREATEOBJECT("SQLDMO.SQLServer")
sBackup = CREATEOBJECT("SQLDMO.Backup2")
sServer.Connect("sServer", "sa", "")
mDatabase  = sServer.databases.count
FOR i=1 TO mdatabase
sDatabase = sServer.Databases.Item(i)
sBackup.Database = sDatabase.Name
sBackup.Files = FileName
sBackup.SQLBackup(sServer)
sServer.DisConnect()
ENDFOR
Catch ex As Exception
MessageBox.Show(ex.Message)
```

```
EndTry
```

16. Как запустить архиватор arj из VFP, чтобы он при запуске не мелькал на экране

Использовать команду `RUN` или ! с ключом `/ N`.

Точно так же можно избежать мелькания Dos-окна и при запуске других программ.

17. Как определить серийный номер жесткого диска

Используйте API-функцию `GetVolumeInformation` (листинг 28.17).

```
DECLARE Integer GetVolumeInformation In Win32Api;
  String @ Rootn, String @ Voln, Integer V_Lenn,;
  Integer @ SNn, Integer @ kmn, Integer @ Fln,;
  String @ Sys_fn, Integer S_Lenn
  Rootn='c:\'
  Voln='**********'
  V_lenn=10
  snn=10
  kmn=10
  Fln=10
  Sys_fn='**********'
  s_lenn=10
  =GetVolumeInformation ;
  (@Rootn,@Voln,V_Lenn,@SNn,@kmn,@Fln,@Sys_fn,@S_Lenn)
  l_sern=SNn
  if l_sern<0
        l_sern=4294967296+l_sern
  ENDIF
  ? l_sern
```

18. Отображение текста на форме в виде бегущей строки

Для получения эффекта "бегущей строки" на форме нам понадобятся два элемента управления: `Label` и `Timer`. Наши действия по отображению "бегущей строки" будут заключаться в следующем:

1. Создаем форму.
2. Размещаем на ней элемент управления `Label`.
3. Размещаем таймер.
4. `Label`: в свойстве `Caption` размещаем текст, который желаем вывести в качестве "бегущей строки", устанавливаем необходимый вид и размер шрифта. Для того

чтобы текст появлялся из-за правой границы формы, установим значение в свойстве `Left` равным `ThisForm.Width`.

5. `Timer`: в свойстве `Interval` установим значение 20 (подберите значение на свой вкус). В методе `Timer` помещаем следующий код:

```
ThisForm.label1.Left = ThisForm.label1.Left – 1
IF ThisForm.label1.Left = -ThisForm.label1.Width
     ThisForm.label1.Left=ThisForm.Width
ENDIF
```

Вот и все, собственно. Запускайте и смотрите. На компакт-диске демо-пример находится в CHAR28\28_17_dop\beg.scx.

19. Создание каталога для отчетов и создание ссылки на него на рабочем столе пользователя

Код создания каталога для отчетов ссылки на него на рабочем столе приведен в листинге 28.18.

Листинг 28.18. Создание каталога для отчетов и создание ссылки на него на рабочем столе пользователя

```
WSHShell = CREATEOBJECT("WScript.Shell")
  lcPath= WSHShell.SpecialFolders("MyDocuments")+"\REPORTS\"
  IF !DIRECTORY(lcPath)
        MKDIR (lcPath)
  ENDIF

  lcFile= WSHShell.SpecialFolders("Desktop")+'\Reports.lnk'
  IF !FILE(lcFile)
        loShortcut = WSHShell.CreateShortcut(lcFile)
        loShortcut.TargetPath = lcPath
        loShortcut.WindowStyle = 4
        loShortcut.SAVE()
  ENDIF
```

20. Определение информации о дисках компьютера

Можно использовать, например, `FileSystemObject` — объект нового языка Microsoft® Visual Basic® Scripting Edition.

Microsoft® Visual Basic® Scripting Edition — это новейший член семейства языков программирования Visual Basic. Использование Visual Basic Scripting объектов в VFP предоставляет нам несколько новых возможностей. Наибольший интерес представляет такой мощный объект, как `FileSystemObject`. С помощью этого объекта мы можем достаточно просто и быстро получить информацию о дисководах, каталогах и файлах, работать с каталогами и файлами, т. е. создавать, удалять, перемещать, менять их атрибуты и т. д. (листинг 28.19).

```
CLEAR
LOCAL loFSO, loDrivesCol, loDrive, m.pathto
loFSO       = CREATEOBJECT('Scripting.FileSystemObject')
loDrivesCol = loFSO.Drives
FOR EACH loDrive IN loDrivesCol
      ? loDrive.DriveLetter
      ?? CHR(9)
      ?? loDrive.DRIVETYPE
      ?? CHR(9)
      ?? loDrive.IsReady
NEXT
```

Существует и более простой, но оттого не менее рабочий вариант:

```
IF DISKSPACE("a:") = -1 &&нет дискеты
MESSAGEBOX("Вставьте дискету в дисковод",48,"Please...")
RETURN .F.
ENDIF
```

21. Создание "на лету" группы кнопок или переключателей (из таблицы)

Иногда группа кнопок или переключателей не может быть размещена на форме с помощью Конструктора форм. Это происходит в случае, когда количество кнопок или переключателей заранее неизвестно. Например, наименования кнопок могут храниться в таблице или курсоре. В этом случае применяется формирование `CommandGroup` или `OptionGroup` "на лету", программным способом (листинг 28.20). В примере приведено формирование меню с помощью `CommandGroup`.

1. Формируем курсор из данных таблицы, затем добавляем в него дополнительные пункты меню.
2. Создаем класс, содержащий объект `CommandGroup` с одной кнопкой, с помощью команды.

   ```
   DEFINE CLASS newmenu As Form...
   ```

3. В событии `Init` подсчитываем количество кнопок меню и определяем их свойства.
4. В событии `Click` вводим обработку кнопок, описываем, что же должно происходить при нажатии этой конкретной кнопки.
5. В событии `Destroy` освобождаем меню.

```
SELECT WAY.NOM, W, NAME FROM WAY INTO CURSOR WAYS READWRITE
INSERT INTO ways (NAME) VALUES ("Сводный по участкам")
INSERT INTO ways (NAME) VALUES ("Формирование посылки")
INSERT INTO ways (NAME) VALUES ("Архивирование посылки")
INSERT INTO ways (NAME) VALUES ('Выход')
```

```
oMENU = NEWOBJECT('newmenu')
oMENU.SHOW()
oMENU.VISIBLE = .T.
READ EVENTS
CANCEL
**
DEFINE CLASS newmenu AS form
HEIGHT = 600
WIDTH = 451
SHOWWINDOW = 2
DOCREATE = .T.
CAPTION = "Выберите участок"
NAME = "newmenu"
ADD OBJECT COMMANDGROUP1 AS COMMANDGROUP WITH AUTOSIZE = .T.,;
 BUTTONCOUNT = 2,;
 ANCHOR = 0, ;
 VALUE = 1,;
 HEIGHT = 64,;
 LEFT = 98, ;
 TOP = 0, ;
 WIDTH = 246,;
 NAME = "Commandgroup1",;
 COMMAND1.AUTOSIZE = .F.,;
 COMMAND1.TOP = 5,;
 COMMAND1.LEFT = 5, ;
 COMMAND1.HEIGHT = 23,;
 COMMAND1.WIDTH = 235, ;
 cOMMAND1.PICTURE = "book.bmp",;
 COMMAND1.CAPTION = "Command1",;
 COMMAND1.STYLE = 0, ;
 COMMAND1.PICTUREPOSITION = 0, ;
 COMMAND1.PICTUREMARGIN = 5, ;
 COMMAND1.PICTURESPACING = 5, ;
 COMMAND1.ALIGNMENT = 0, ;
 COMMAND1.NAME = "Command1"
 *

PROCEDURE Init
Public KOLVO, KOLVO1
SELECT WAY
COUNT FOR ! DELETED() TO KOLVO1
SELECT WAYS
COUNT FOR ! DELETED() TO KOLVO
SELECT WAYS
GOTO TOP
THISFORM.COMMANDGROUP1.BUTTONCOUNT = KOLVO
THISFORM.COMMANDGROUP1.AUTOSIZE = .f.
THISFORM.COMMANDGROUP1.LEFT = THISFORM.WIDTH/2+100
THISFORM.WINDOWSTATE = 2
SELECT WAYS
GOTO TOP
FOR i=1 TO kolvo
 lcNameOB = "Command"+TRANSFORM(i)
 this.addobject(lcNameOB ,"CommandGroup")
 with this.CommandGroup1.Objects(i)
```

```
.caption=IIF(!EMPTY(ways.nom),STR(ways.nom,2)+' —
'+cpconvert(866,1251,ALLTRIM(ways.name)),ALLTRIM(ways.name))
.width=235
.height=23
.picture='book.bmp'
.picureposition=0
.alignment=0
.picturespacing=15
.picturemargin=5
endwith
SELECT WAYS
SKIP
ENDFOR
ENDPROC
**
PROCEDURE commandgroup1.Click
DO CASE
   CASE THISFORM.COMMANDGROUP1.VALUE<=KOLVO1
MessageBox('Отсутствует форма для обработки')
  CASE THISFORM.COMMANDGROUP1.VALUE=KOLVO1+1
MessageBox('Отсутствует форма для обработки')
  CASE THISFORM.COMMANDGROUP1.VALUE=KOLVO1+2
MessageBox('Отсутствует форма для обработки')
CASE THISFORM.COMMANDGROUP1.VALUE=KOLVO1+3
MessageBox('Отсутствует форма для обработки')
CASE THISFORM.COMMANDGROUP1.VALUE=KOLVO1+4
        THISFORM.DESTROY()
ENDCASE
ENDPROC
**
PROCEDURE destroy
CLEAR EVENTS
ENDPROC
ENDDEFINE
```

22. При нажатии клавиши <Enter> первая колонка *Grid* скрывается за границу *Grid*

Если `Grid` имеет много колонок, не все они помещаются на экране. При передвижении по колонкам с помощью клавиши <Enter> можно наблюдать такой эффект: самые левые колонки исчезают за левой границей `Grid`, давая вам возможность просматривать следующие колонки. А как быть, если такой визуальный эффект вам не нужен? Использовать `NODEFAULT` в событии `KeyPress`? Это не всегда возможно, потому что в `KeyPress` может находиться обработка каких-то комбинаций клавиш. Самый простой выход из создавшейся ситуации — "заморозить" ту колонку, которая не должна двигаться. В событии `Init() Grid` поместите код (листинг 28.21).

```
this.LockColumns=1
this.LockColumnsLeft=1
```

23. Контекстный поиск в *Grid*

В событии `KeyPress() Grid1.Text1` введите код (листинг 28.22).

```
thisform.keypreview=.t.
thisform.grid1.column1.header1.caption=ss
** аналогично можно вывести строку контекстного поиска в
** этикетку (Label)
** thisform.label1.caption=ss
SEEK (PROPER(ss))
this.Refresh()
```

Возможны и другие варианты, например, использование класса `InGrid` или других подобных.

24. Информирование пользователя о выполнении какого-либо процесса

Воспользуемся API-функцией (листинг 28.23).

```
DECLARE INTEGER NetMessageBufferSend IN NETAPI32 STRING, STRING, STRING, STRING,
INTEGER
  lcTo = "Кому_послать" && Аналогично параметру net send — имя компьютера или имя
пользователя (в домене)
  lcMessage = "текст сообщения"
  lnRes = NetMessageBufferSend(.Null., ;
      STRCONV(m.lcTo + CHR(0), 5), .Null., ;
      STRCONV(m.lcMessage + CHR(0), 5), LEN(m.lcMessage) * 2)
  IF m.lnRes # 0
      ? "Ошибка # ", m.lnRes
  ENDIF
```

25. Как вернуть название месяца по его номеру

Текст кода приведен в листинге 28.24.

```
procedure c_month
  parameters m.data
  return GetWordNum('январь февраль март апрель май июнь июль август сентябрь октябрь
ноябрь декабрь',Month(m.Data))
```

26. Как выгрузить неактивную программу у пользователей

Автор программы дистанционной выгрузки — Владимир Трухин.

Приложения создаются для пользователей и запускаются пользователями. Будучи запущенным пользователем, приложение может работать с утра и до вечера. Пользователь в это время может беззаботно уйти обедать или совещаться в смежный отдел.

Но вот незадача, именно в это самое время вам, как администратору системы или как разработчику, необходимо изменить структуры таблиц или выполнить срочные работы на сервере. Как быть с работающим приложением? Сбросить соединение пользователя или отключить сервер, несмотря на работающее приложение? Это может привести к потере данных. Дождаться появления пользователя? Но от этого могут пострадать другие клиенты, ожидающие завершения работ. Сбегать самому и выгрузить программу? Хорошая идея, если для этого не надо преодолевать десятки километров. Извечный вопрос: "Что делать?".

Решение очевидно. Нужно обеспечить механизм отправки команды администратором, получения этой команды приложением и отработки завершения.

Нет, это не оригинальная идея. Вы могли читать о том, что администратору достаточно разместить в некотором каталоге некоторый файл, а приложению обнаружить этот файл и завершиться без участия пользователя.

Все это было бы очень хорошо, если бы не следующие моменты:

- каждое приложение имеет свою специфику и должно выгружаться особенным для него способом. Как сделать так, чтобы в рамках одного класса обеспечить настройку завершения конкретного приложения?
- система может состоять из некоторого числа связанных или несвязанных приложений, поэтому должна существовать возможность выгрузки выбранного приложения или всех приложений разом;
- приложения не должны внезапно исчезать с экрана;
- пользователь должен получить сообщение о причинах выгрузки приложения или предполагаемом времени возобновления работы;
- пользователь должен иметь отсрочку выгрузки приложения, чтобы успеть произвести ручную выгрузку приложения так, как он это делает обычно. При этом он должен видеть, сколько времени осталось до автоматического завершения.

Для реализации этих требований средство автоматической выгрузки приложения должно обладать следующими возможностями:

- объект класса автоматической выгрузки приложения должен быть невидимым до получения команды выгрузки приложения;
- объект класса должен реагировать на два события:

  - появление в каталоге файла, являющегося командой выгрузки этого приложения;
  - появление в каталоге файла, являющегося командой выгрузки всех приложений;
- ♦ объект класса должен хранить ссылку на процедуру завершения приложения, имитирующую действия пользователя по завершении приложения;
- ♦ при получении команды выгрузки приложения объект класса должен стать видимым;
- ♦ командный файл должен содержать текст сообщения, которое считывается объектом класса и показывается пользователю;
- ♦ на экземпляре класса должен располагаться индикатор прогрессии, показывающий время, оставшееся до полного завершения приложения;
- ♦ по истечении времени ожидания действий пользователя, экземпляр класса должен запустить процедуру завершения приложения;
- ♦ экземпляр класса должен работать как в приложении с главным окном, так и в приложении без главного окна.

Индикатор прогрессии

В качестве индикатора прогрессии можно использовать любой имеющийся класс индикатора процесса или элемент ActiveX. Предположим, что у нас нет ни того, ни другого. Создать индикатор процесса совсем не трудно. Достаточно использовать контейнер, как базовый класс, и разместить в нем объекты, изображающие индикатор. Это могут быть объекты класса SHAPE. Первый будет показывать общую длину процесса, другой текущее значение процесса.

Определение такого класса может выглядеть таким образом, как в листинге 28.25.

```
DEFINE CLASS progressbar AS container
** Длина полосы индикатора
Width = 100
** Высота полосы индикатора
Height = 20
BackStyle = 0
BorderWidth = 0
** Максимальное значение переменной процесса
MaxValue = (this.Width)
** Текущее значение переменной процесса
CurrentValue = 0
Name = "progressbar"
** Объект, показывающий общую длину процесса
ADD OBJECT border AS shape WITH ;
** Разместить на всей площади контейнера
Top = 0, ;
```

```
Left = 0, ;
Height = (this.parent.Height), ;
Width = (this.parent.Width), ;
Name = "Border"
ADD OBJECT bar AS shape WITH ;
** Разместить по высоте контейнера с нулевой длиной
Top = 0, ;
Left = 0, ;
Height = (this.parent.Height), ;
Width = 0, ;
BorderStyle = 0, ;
BorderWidth = 0, ;
Curvature = 0, ;
FillStyle = 0, ;
FillColor = RGB(0,0,255), ;
Name = "Bar"
PROCEDURE currentvalue_assign
    ** При изменении текущего значения параметра
    ** необходимо перерисовать объект текущего значения
    LPARAMETERS vNewVal
    ** Анализ значения принятого параметра
    do case
        case m.vNewVal<0
            THIS.CurrentValue = 0
        case m.vNewVal>0 and m.vNewVal<=THIS.MaxValue
            THIS.CurrentValue = m.vNewVal
        case m.vNewVal>THIS.MaxValue
            THIS.CurrentValue = THIS.MaxValue
    endcase
    перерисовка объекта
    THIS.Bar.Width=int(THIS.Width*(THIS.CurrentValue/THIS.MaxValue))
ENDPROC
PROCEDURE Init
    ** Настройка объектов на размер контейнера
    this.Bar.Height=this.Height
    this.Bar.Width=0
    this.Border.Height=this.Height
    this.Border.Width=this.Width
ENDPROC
ENDDEFINE
```

Окно оповещения пользователя

Окно оповещения пользователя будет появляться на экране после обнаружения команды выгрузки приложения. На этом окне размещается индикатор прогресса, текст извещения для пользователя и мигающий красный индикатор, который можно использовать для привлечения внимания пользователя.

После появления этого окна на экране должен запуститься индикатор процесса. Когда индикатор достигнет своего граничного значения, должна запуститься процедура завершения приложения.

Примерный код для этого класса (листинг 28.26).

```
DEFINE CLASS alarm AS form
Height = 200
Width = 300
Desktop = .T.
DoCreate = .T.
AutoCenter = .T.
BorderStyle = 2
Caption = "Завершение работы приложения"
TitleBar = 1
AlwaysOnTop = .T.
BackColor = RGB(255,255,128)
MaxValue = (this.Progress.Width)
CurrentValue = 0
lightFlash = 0
Name = "alarm"
ParentRef = .F.

    ADD OBJECT label1 AS label WITH ;
    FontBold = .F., ;
    FontSize = 9, ;
    WordWrap = .T., ;
    Alignment = 2, ;
    BackStyle = 0, ;
    Caption = "Внимание! "+;
              "Приложение будет завершено. "+;
              "Сохраните данные и немедленно выйдите из программы!", ;
    Height = 33, ;
    Left = 26, ;
    Top = 5, ;
    Width = 265, ;
    Name = "Label1"

** Таймер времени ожидания
ADD OBJECT timer AS timer WITH ;
    Top = 11, ;
    Left = 270, ;
    Height = 23, ;
    Width = 23, ;
    Enabled = .F., ;
    Interval = 500, ;
    Name = "Timer"

** Объект индикатора процесса
ADD OBJECT progress AS progressbar WITH ;
         Top = 45, ;
         Left = 6, ;
         Width = 285, ;
         Height = 20, ;
         Name = "Progress", ;
         Border.DefHeight = "", ;
         Border.DefWidth = "", ;
         Border.BackStyle = 0, ;
         Border.Name = "Border", ;
```

```
    Bar.DefHeight = "", ;
    Bar.Name = "Bar"
ADD OBJECT inform AS editbox WITH ;
    FontBold = .F., ;
    Alignment = 0, ;
    BackStyle = 1, ;
    BorderStyle = 1, ;
    Enabled = .F., ;
    Height = 138, ;
    Left = 6, ;
    ReadOnly = .F., ;
    Top = 78, ;
    Width = 288, ;
    DisabledBackColor = RGB(255,255,128), ;
    DisabledForeColor = RGB(0,0,0), ;
    Name = "Inform"
PROCEDURE maxvalue_access
     RETURN THIS.Progress.MaxValue
ENDPROC
PROCEDURE maxvalue_assign
    LPARAMETERS vNewVal
    THIS.Progress.MaxValue = m.vNewVal
ENDPROC
PROCEDURE start
    this.Timer.Enabled=.T.
ENDPROC
PROCEDURE getcaption
    if type('gcVersion')='C'
        return allt(gcVersion)
    else
        return 'Внимание!'
    endif
ENDPROC
PROCEDURE Init
    LPARAMETERS lnTimeOut
    if type('lnTimeOut')!='N'
        this.MaxValue=lnTimeOut
    endif
ENDPROC
PROCEDURE timer.Timer
    this.parent.LightOn.Visible=!this.parent.LightOn.Visible
    this.parent.LightOff.Visible=!this.parent.LightOff.Visible
    local lnLightFlash
    lnLightFlash=this.parent.LightFlash
    this.parent.LightFlash=this.parent.LightFlash+1
    this.parent.CurrentValue=this.parent.CurrentValue+this.Interval
    if this.parent.CurrentValue>=this.parent.MaxValue
        local loShutDown
        loShutDown=this.parent.ParentRef
        loShutDown.TimeOut()
        this.parent.Release()
    endif
ENDPROC
PROCEDURE inform.Init
    this.Value='Приложение будет выгружено для изменения!'     ENDPROC
```

```
ENDDEFINE
```

Окно оповещения пользователя в приложении на базе формы верхнего уровня

Определим подкласс для класса окна оповещения пользователя. Цель этого подкласса — работа в приложении на базе формы верхнего уровня без главного окна Visual FoxPro (листинг 28.27).

```
DEFINE CLASS alarmwindowtlf AS alarmwindow
     Desktop = .F.
     ShowWindow = 1
     Name = "alarmwindowtlf"
     Label1.Name = "Label1"
     LightOn.Height = 12
     LightOn.Width = 12
     LightOn.Name = "LightOn"
     LightOff.Height = 12
     LightOff.Width = 12
     LightOff.Name = "LightOff"
     Timer.Name = "Timer"
     PROGRESS.Border.DefHeight = ""
     PROGRESS.Border.DefWidth = ""
     PROGRESS.Border.Name = "Border"
     PROGRESS.Bar.DefHeight = ""
     PROGRESS.Bar.Name = "Bar"
     PROGRESS.Name = "PROGRESS"
ENDDEFINE
```

Класс дистанционной выгрузки приложения

Определим класс дистанционной выгрузки приложения. Именно он должен невидимо работать все время существования приложения, обнаруживать командные файлы, запускать окно оповещения пользователя и в случае необходимости самостоятельно выгружать приложение (листинг 28.28).

```
DEFINE CLASS powershutdownms AS timer
     Height = 23
     Width = 23
     Enabled = .F.

** Интервал поиска файла (по умолчанию)
     Interval = 60000

** Имя файла для выгрузки всех приложений
systemshutdownfile = "ShutDown.txt"
```

```
** Имя файла для выгрузки конкретного приложения
     taskshutdownfile = NULL

** Время ожидания действий пользователя
     shutdownwaittime = 180000

** Имя процедуры завершения приложения
     shutdownhandler = NULL

     PROTECTED alarmmess
     alarmmess = NULL

     Name = "powershutdownms"

     PROTECTED inform

** Запуск окна оповещения пользователя
     PROCEDURE startalarm
         local loAlarm
         this.AlarmMess=createobject('AlarmWindow')
         loAlarm=this.AlarmMess
         loAlarm.Inform.Value=this.Inform
         loAlarm.MaxValue=this.ShutDownWaitTime
         loAlarm.ParentRef=this
         loAlarm.Show()
         loAlarm.Start()
     ENDPROC
** Обработка события завершения ожидания пользователя
     PROCEDURE timeout
         if !isnull(this.AlarmMess)
             local loAlarmMess
             loAlarmMess=this.AlarmMess
             if !isnull(this.ShutDownHandler)
                 if type('this.ShutDownHandler')='C'
                     local lcShutDownHandler
                     lcShutDownHandler=this.ShutDownHandler
                     do &lcShutDownHandler
                 else
                     this.ShutDown()
                 endif
             else
                 this.ShutDown()
             endif
         endif
     ENDPROC
** Завершение приложения (по умолчанию)
     PROCEDURE shutdown
         quit
     ENDPROC
** Поиск командного файла
     PROCEDURE Timer
         local llShutDown
         llShutDown=.F.
         if file(this.SystemShutDownFile)
             this.Inform=filetostr(this.SystemShutDownFile)
```

```
            llShutDown=.T.
        else
            if !isnull(this.TaskShutDownFile)
                if type('this.TaskShutDownFile')='C'
                    if file(this.TaskShutDownFile)
                        this.Inform=filetostr(this.TaskShutDownFile)
                        llShutDown=.T.
                    endif
                endif
            endif
        endif
        if llShutDown
            this.Enabled=.F.
            this.StartAlarm()
        endif
    ENDPROC
    PROCEDURE Init
        this.Inform=''
    ENDPROC
ENDDEFINE
```

Подкласс класса дистанционной выгрузки приложения для работы без главного окна VFP

Этот подкласс будет отличаться от своего родителя только кодом метода запуска окна оповещения пользователя — `StartAlarm` (листинг 28.29).

Листинг 28.29. Подкласс класса дистанционной выгрузки для работы без главного окна VFP

```
DEFINE CLASS powershutdowntlf AS powershutdownms
    Name = "powershutdowntlf"
    PROCEDURE startalarm
        local loAlarm
        this.AlarmMess=createobject('AlarmWindowTLF')
        loAlarm=this.AlarmMess
        loAlarm.Inform.Value=this.Inform
        loAlarm.MaxValue=this.ShutDownWaitTime
        loAlarm.ParentRef=this
        loAlarm.Show()
        loAlarm.Start()
    ENDPROC
ENDDEFINE
```

Разместив код для создания объекта этого класса в MAIN-модуле приложения, мы обеспечим дистанционную выгрузку приложения и развяжем себе руки.

Пример такого кода может выглядеть следующим образом (листинг 28.30).

```
** MAIN
PROCEDURE MAIN
PUBLIC goPSD
```

```
SET CLASSLIB TO SHUTDOWN
goPSD=createobject('PowerShutDownMS')
** Интервал для поиска командного файла = 3 минут
goPSD.Interval=3*60*1000
** Время ожидания действий пользователя = 5 минут
goPSD.ShutDownWaitTime=5*60*1000
** Процедура завершения
goPSD.ShutDownHandler='Finish'
goPSD.Enabled=.T.
** ...
** код инициализации приложения
** ...
ENDPROC

PROCEDURE Finish
** код завершения приложения
ENDPROC
```

Демонстрационный пример находится на компакт-диске в CHAR28\Shutdown.

27. Как разместить на форме часы

Код приведен в листинге 28.31.

```
PUBLIC oform1
  oform1=NEWOBJECT("form1")
  oform1.Show
  RETURN
  DEFINE CLASS form1 AS form

        Height = 83
        Width = 192
        AutoCenter = .T.
        BorderStyle = 2
        Caption = "Form1"
        FontName = "MS Sans Serif"
        FontSize = 8
        MaxButton = .F.
        MinButton = .F.
        Icon = _Screen.Icon
        Name = "Form1"

        ADD OBJECT label1 AS label WITH ;
                Caption = "", ;
                Height = 17, ;
                Left = 10, ;
                Top = 15, ;
                Width = 53, ;
                Name = "Label1"

        ADD OBJECT timer1 AS timer WITH ;
                Top = 45, ;
```

```
                Left = 10, ;
                Height = 23, ;
                Width = 23, ;
                Interval = 1000, ;
                Name = "Timer1"

        PROCEDURE Init
                thisform.timer1.Timer
        ENDPROC
        PROCEDURE timer1.Timer
                thisform.label1.Caption=Time()
        ENDPROC
ENDDEFINE
```

28. Как заставить цифровую клавиатуру отображать точку при любой раскладке (Ru/En)

На альтернативной цифровой клавиатуре (правая часть клавиатуры) под клавишей <NumLock> есть клавиша с "точкой". Однако она вводит именно точку только при английской раскладке клавиатуры, а при русской раскладке клавиатуры вводит запятую. При вводе чисел с дробной частью это создает проблемы. Как можно заставить вводить символ точки вне зависимости от текущей раскладки клавиатуры? В событии `KeyPress` для вашего объекта `TextBox` запишите такой код (листинг 28.32).

```
LPARAMETERS nKeyCode, nShiftAltCtrl
IF InList(CHR(nKeyCode),'.',',') AND CHR(nKeyCode)<>SET('POINT')
      NODEFAULT
      KeyBoard SET('POINT') PLAIN CLEAR
ENDIF
```

Теперь при попытке ввода точки или запятой будет автоматически введен символ, который вы приняли в качестве разделителя целой и дробной части. Даже если это, например, дефис.

29. Как заменить сообщение "Invalid Date" при вводе некорректной даты

Если в `TextBox` вводится некорректная дата, то появляется сообщение "Invalid Date". Как можно проконтролировать корректность ввода даты до появления этого сообщения и заменить данное сообщение своим? Начиная с версии Visual FoxPro 5, для объекта `TextBox` добавлено свойство `StrictDataEntry`. По умолчанию это свойство имеет значение: 1 — `Strict`. Установите его в значение: 0 — `Loose`.

В случае если введено некорректное значение даты, то никакого сообщения об ошибке вообще не возникает. Просто значение свойства `TextBox.Value` становится пустым.

Это можно определить в событии TextBox.Valid() и выдать собственное сообщение об ошибке (листинг 28.33).

```
* Событие Text1.Valid
 IF EMPTY(This.Value) = .T. AND This.Text <> TRANSFORM({},"@"+This.Format)
       MessageBox("Дата введена некорректно!")
 ENDIF
```

Свойство TextBox.Text — это то, что реально отображено на форме в объекте TextBox, т. е. то, что реально видит пользователь. Вот к проверке на пустой результат добавляется проверка на тот факт, что пользователь что-то ввел. К сожалению, просто проверить это свойство при помощи функции EMPTY() не получится. В нем присутствуют символы-разделители, т. е. этот текст по определению "не пустой".

Более того, данное сравнение не корректно, если в качестве символа разделителя будет использован какой-либо нестандартный символ, заданный через свойство Text1.DateMark. Впрочем, в своей программе вы ведь знаете, какая символьная строка соответствует пустой дате. Вот с ней и сравнивайте значение Text1.Text.

Если объект TextBox не связан ни с каким источником данных, т. е. и TextBox.ControlSource, и TextBox.Value при открытии формы не указаны, то для того, чтобы символы-разделители в дате не съезжали при вводе, необходимо указать нужный формат

```
Text1.Format = "D"
```

30. Как получить программный код создания структуры базы данных?

Вместе с FoxPro поставляется утилита (программа) по генерации программного кода структуры существующей базы данных. Она называется GenDBC.prg. В командном окне FoxPro дайте команду

```
DO HOME()+"Tools\GenDBC\GenDBC.prg"
```

Сначала вам будет предложено указать контейнер базы данных (файл DBC), структуру которого вы хотите получить.

Следующим шагом будет предложено указать имя файла PRG, куда будет записана полученная структура.

В результате работы данной утилиты вы получите файл PRG, который можно просмотреть из среды FoxPro через стандартную команду

```
MODIFY COMMAND
```

Запуск данного файла на исполнение создаст структуру базы данных, полностью идентичную тому контейнеру базы данных, который был использован в качестве образца. Разумеется, пустую, без самих данных.

31. Как перенести данные из текстового файла в таблицу

В отдельных случаях применима команда APPEND FROM, но это для случая, когда текст имеет однотипные разделители.

А следующее решение кажется более универсальным (листинг 28.34).

```
CREATE TABLE C:\Table.dbf FREE (ls N(7),fio C(50),f1 N(4),street N(7),dom
N(7),korp N(7),flat N(7))
  fHandle=FOPEN("C:\TextFile.txt")
  DO WHILE !FEOF(fHandle)
        Stroka=FGETS(fHandle)
        INSERT INTO Table VALUES (;
                VAL(STREXTRACT(Stroka,""," ")),;
                STRE(Stroka,","," "),;
                VAL(STREXTRACT(Stroka," "," ",4)),;
                VAL(STREXTRACT(Stroka," "," ",5)),;
                VAL(STREXTRACT(Stroka," "," ",6)),;
                VAL(STREXTRACT(Stroka," "," ",7)),;
                VAL(STREXTRACT(Stroka," "," ",8)))
  ENDDO
```

32. Получение данных об установленных программных продуктах

Замечательным инструментом разработчика является WMI (Windows Management Instrumentation) — интерфейс, обеспечивающий взаимодействие с компонентами системы. WMI можно использовать для управления компьютерами с помощью сценариев. В частности, с помощью WMI можно получить список программ, установленных посредством Windows Installer. Но не только. WMI — это удобный инструмент системного администратора, позволяющий автоматизировать его работу. Например, с помощью WMI можно создавать назначенные задания, а затем проверять, выполнились ли они в указанное время. Или, например, изменить метку тома (листинги 28.35—28.37).

```
clear
*подключение к пространству имен
objService = GetObject("winmgmts:\\.\Root\CIMV2")
*получение объекта логического диска "K:"
colDisks = objService.ExecQuery("Select * from Win32_LogicalDisk Where DeviceID =
'K:'")
For Each objDisk In colDisks
 ?objDisk.VolumeName
```

```
*смена метки тома
    objDisk.VolumeName = "New"
*запись изменений
    objDisk.Put_
? objDisk.VolumeName
Next
```

Листинг 28.36. Определение списка установленных на компьютере программных продуктов

```
objService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\CIMV2")
LOCAL oCol AS Collection
oCol=objService.ExecQuery("SELECT * FROM Win32_Product")
For Each objProduct IN oCol
*       наименование
        ? objProduct.Name
        ? objProduct.Caption
        ? objProduct.Description
*       производитель
        ? objProduct.Vendor
*       версия
        ? objProduct.Version
*       серийный номер
        ? objProduct.IdentifyingNumber
*       дата установки
        ? objProduct.InstallDate
        ? objProduct.InstallDate2
*       каталог установки
        ? objProduct.InstallLocation
*       **состояние установки
        ? objProduct.InstallState
*       путь к файлу установки MSI
        ? objProduct.PackageCache
endfor
```

```
clear
objService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\CIMV2")
objNewJob=objService.Get("Win32_ScheduledJob")
errJobCreated=objNewJob.Create("Notepad.exe", "********012500.000000-420", .T. , 4, ,
.T., .T.)
         ? errJobCreated
If errJobCreated <> 0
        WAIT WINDOW  "Ошибка при создании задания"
Else
        WAIT window "Задание создано"
EndIf
```

Следует заметить, что перечень классов WMI зависит от реализации операционной системы. Поэтому, чтобы быть уверенным в том, что данный класс существует в данной операционной системе, следует либо создать список классов, либо выполнить

запрос на существование класса. В листинге 28.38 создается таблица WMI_Classes с перечнем классов и их свойств.

```
LOCAL lcComputer, loWMIService, loClass, loQualifier, llIsQualifier
CREATE TABLE wmi_classes (name c(100), Prop c(100))
lcComputer="."
loWMIService = Getobject("winmgmts:\\" + lcComputer + "\root\cimv2")
For Each loClass In loWMIService.SubclassesOf
   If Upper(Left(loClass.Path_.Class,5)) = "WIN32" Then
      For Each loQualifier In loClass.Qualifiers_
         If Upper(Alltrim(loQualifier.Name)) = "ASSOCIATION" Then
            llIsQualifier = .T.
         Else
            llIsQualifier = .F.
         Endif
      Next
      If llIsQualifier = .F.
           ? loClass.Path_.Class
           INSERT INTO wmi_classes (NAME) values (loClass.Path_.Class)
           loWMIClass = loWMIService.Get(loClass.Path_.Class)
       For Each loProperty in loWMIClass.Properties_
          ?  loProperty.name
          INSERT INTO wmi_classes (NAME, prop) values
(loClass.Path_.Class,loProperty.name)
     EndFor
   Endif
   Endif
EndFor
BROWSE
```

Синтаксис запроса составляется следующим образом:

```
<имя объекта/коллекции объектов> = loWMIService.ExecQuery ;
("Select * from <имя класса WMI>")
```

Здесь `имя объекта/коллекции объектов` представляет собой стандартную VFP переменную, например: `loPrinters` или `loSharedDisks`.

Таким образом, например, вы можете получить информацию о существующих назначенных заданиях на указанном компьютере. Учтите, что WMI создает список назначенных заданий, созданных с помощью WIN32_SheduledJob или утилиты At.exe. В список не включаются задания, созданные вручную с помощью Task Sheduler (листинг 28.39).

```
clear
objService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\CIMV2")
LOCAL oCol AS Collection
oCol=objService.ExecQuery("SELECT * FROM Win32_ScheduledJob")
For Each objJob IN oCol
```

```
   ? "Caption: "      +objJob.Caption && Наименование
   ? "Command: "      +objJob.Command && Команда
   ? "Days Of Month: "+TRANSFORM(objJob.DaysOfMonth) && День месяца
   ? "Days Of Week: " +TRANSFORM(objJob.DaysOfWeek)   && День недели
   ? "Elapsed Time: " +TRANSFORM(objJob.ElapsedTime)  && Время выполнения
   ? "Job Status: "   +TRANSFORM(objJob.JobStatus)       && Статус
                                                              Выполнения
   ? "================================================"
EndFor
```

33. Из какого каталога запущена программа (EXE-файл)?

Существуют функции, которые могут решить подобную проблему *при определенных условиях*. Надо всегда помнить, что эти функции имеют ограничения.

```
* Путь доступа к первой запущенной программе (главному файлу проекта)
  JustPath(SYS(16,0))

 * Путь доступа к текущей директории
  FullPath("") — пустая строка (две кавычки подряд) — обязательны
  SYS(5)+SYS(2003)
  SET("DEFAULT")+CURDIR()
  _VFP.DefaultFilePath
```

34. VFP+Internet Explorer

Текст кода приведен в листинге 28.40.

```
oleApp = CreateObject("InternetExplorer.Application")
oleApp.Visible=.t.
oleApp.Navigate("http://www.foxclub.ru")
```

35. Visual FoxPro + Lotus Notes

Текст кода предоставлен Николаем Кремко (Минск) (листинг 28.41).

```
clear
lotus = Createobject( "FOX2ForLotus" )
lotus.Initialize()
lotus.SendMail("Nikolai Kremko","1","")
lotus.StoreAllDocAttachments("c:\temp", "1006931_1006932","","","",1,{},{},0)
? lotus.mn_Files

  Define Class FOX2ForLotus As Line
```

```
mo_Session =.Null.
mo_DB =.Null.
mo_Mail=.Null.
mc_PassWord=""
mn_Files= 0
mt_DocTimeStart = {//,:} && \Пустое значение
mt_DocTimeEnd    = {^2020-12-31,0}

        Function Initialize
This.mo_Session =Createobject("lotus.NotesSession")
 *Логин возможен 2-мя способами (Зависит от настроек лотуса.
* У меня возможен только второй, поэтому первый оставлю только в
* комментарии)
 *this.mo_Session.InitializeUsingNotesUserName(<USER>,<PASSWORD>)
If Empty(This.mc_PassWord)
        This.mo_Session.Initialize()
Else
        This.mo_Session.Initialize(This.mc_PassWord)
Endif
This.mo_DB = This.mo_Session.GetDbDirectory("")
This.mo_Mail = This.mo_DB.OpenMailDatabase()
Return 1
Endfunc

        Function StoreAllDocAttachments(tc_Folder,
tc_SubjectFilter,tc_FromFilter,tc_SendToFilter,tc_FilterForFileName, tn_RemoveDoc,
td_DateBegin, td_DateEnd,tn_lookInDeleted)
                && Сохраняет все аттачменты (вложенные в письмо файлы) из всех
писем, по условиям фильтров в tc_Folder
                && Удаляет письма, если все аттачменты удачно записаны в каталог
tc_Folder и tn_RemoveDoc =1
                &&  Возвращает 1, если при этом не было ошибок
                Local i,ii,iii && переменные для циклов
Local lFlag,lflagFrom ,lflagSendTo,ll_LookingInDeleted &&Флаги
tn_lookInDeleted = IF(EMPTY(tn_lookInDeleted),0,tn_lookInDeleted)
LOCAL Subj,allDoc,Doc &&
With This
External Array items,aFrom,aSendTo
allDoc = This.mo_Mail.AllDocuments
For i=1 To This.mo_Mail.AllDocuments.Count
Doc=allDoc.GetNthDocument(i)
Subj=Doc.GetItemValue("Subject")
lFlag = .T.
If !Empty(tc_SubjectFilter)
If Atc(UPPER(ALLTRIM(tc_SubjectFilter)) ,UPPER( Subj))=0
        lFlag=.F.
Endif
Endif
If lFlag
        aFrom = Doc.GetItemValue("From")
If !Empty(tc_FromFilter)
        lflagFrom = .F.
        For ii=1 To Alen(aFrom)
        If !Empty(aFrom[ii])
```

```
If (Atc(UPPER(ALLTRIM(tc_FromFilter)) , UPPER(aFrom[ii]))  >0 )
        lflagFrom = .T.
Endif
Else
If Atc(UPPER(ALLTRIM(tc_FromFilter)),UPPER(This.mo_Session.UserName))>0
        lflagFrom = .T.
Endif
Endif
Next
Else
        lflagFrom = .T.
Endif
aSendTo = Doc.GetItemValue("SendTo")
If !Empty(tc_SendToFilter)
        lflagSendTo = .F.
        For ii=1 To Alen(aSendTo)
If Atc(UPPER(ALLTRIM(tc_SendToFilter)) ,UPPER( aSendTo[ii]) )  >0
        lflagSendTo = .T.
Endif
Next
Else
        lflagSendTo = .T.
Endif
ENDIF
ll_LookingInDeleted = tn_lookInDeleted=1
If      lFlag And lflagFrom And lflagSendTo
        Doc=allDoc.GetNthDocument(i)
IF (!doc.isDeleted) OR (doc.isDeleted AND ll_LookingInDeleted=.t.)
This.StoreDocAttachmentsToFolder(Doc,tc_Folder,tc_FilterForFileName,tn_RemoveDoc)
ENDIF
ENDIF
Next
Endwith
Return 1
Endfunc
        Function
StoreDocAttachmentsToFolder(to_doc,tc_Folder,tc_FilterForFileName,tn_RemoveDoc)
        && Сохраняет все аттачменты (вложенные в письмо файлы) из документа to_doc
,фильтров в tc_Folder
        && Удаляет письма, если все аттачменты удачно записаны в каталог tc_Folder и
tn_RemoveDoc =1
        &&  Возвращает количество записанных если при этом не было ошибок
External Array items,aFile
Local i,ii,iii
iii=0
If Vartype(to_doc) = "O" And to_doc.IsValid
        items=to_doc.items
        For i = 1 To Alen(items)
        If items[i].Name =="$FILE" Then
                aFile = items[i].Values
                For ii = 1 To Alen(aFile)
        ofile = to_doc.GetAttachment(aFile[ii])
IF EMPTY(UPPER(ALLTRIM(tc_FilterForFileName)))
        ofile.ExtractFile(Addbs(tc_Folder)+aFile[ii])
```

```
        this.mn_Files =  this.mn_Files + 1
ELSE
IF Atc(UPPER(ALLTRIM(tc_FilterForFileName)) ,UPPER( aFile[ii] ))  >0
        ofile.ExtractFile(Addbs(tc_Folder)+aFile[ii])
        this.mn_Files =  this.mn_Files + 1
ELSE
        tl_RemoveDoc = 0
ENDIF
ENDIF
Next
Endif
Next
ENDIF
IF tn_RemoveDoc = 1
        to_doc.Remove(.t.)
ENDIF
Return this.mn_Files
Endfunc
        Function SendMail(tc_adress,tc_Subject,tc_Body,tc_FileName)
                && Отправляет письмо с темой tc_Subject и телом письма tc_Body, и
вложенным файлом tc_FileName
                && от имени текущего пользователя
                &&  Возвращает 1 если при этом не было ошибок
                lo_Doc=This.mo_Mail.CreateDocument()
                lo_Doc.REPLACEITEMVALUE("SendTo", tc_adress)
                lo_Doc.REPLACEITEMVALUE("Form", "Memo")
                lo_Doc.REPLACEITEMVALUE("Subject",tc_Subject)
                lo_RtItem = lo_Doc.CREATERICHTEXTITEM("Body")
                lo_RtItem.APPENDTEXT (tc_Body)
                IF ! EMPTY(tc_FileName)
                        lo_RtItem.EmbedObject(1454,"",tc_FileName)
                ENDIF
                        lo_Doc.SAVEMESSAGEONSEND=1
                        lo_Doc.Send(0)
                RETURN 1
        ENDFUNC
  ENDDEFINE
```