

Библиотека программ VFP2C32

Версия 2.0.0.42

01.05.2024

VFP2C32



VFP2C32 (аббревиатура от "VFP to C 32 bit") — это мост между Visual Foxpro и языком программирования C.

Кроме того, он содержит оболочки вокруг многих функций Windows API.

С помощью этой библиотеки вы можете создавать любые типы данных C — структуры, объединения, массивы — из VFP.

Создание классов-оболочек для типов структур или объединений C объясняется в теме [VFP2C32Front](#).

Массивы C можно создавать с помощью классов в файле vfp2carray.prg.

Оболочки Windows API позволяют использовать широкий спектр встроенных в Windows функций.

API реестра, запрос информации о принтере и управление службами Windows — вот лишь некоторые из них. Чтобы получить обзор, ознакомьтесь с темой [Функции по категориям](#).

Обработка ошибок

Библиотека выдает следующие ошибки.

Номер ошибки	Сообщение	Причина
9	Несоответствие типов данных.	Один из параметров имеет не ожидаемый тип данных.
11	Недопустимое значение, тип или количество аргументов функции.	Один из параметров недействителен или количество параметров неверно.
43	Недостаточно памяти для завершения этой операции.	Не удалось выделить память.
1754	Невозможно найти точку входа.	Функция недоступна в текущей операционной системе.
1098	Содержит системное или пользовательское сообщение об ошибке.	Зависит от контекста - вызовите AErrorEx , чтобы получить расширенную информацию об ошибке.

Распространение

Пакет VFP2C32

Дистрибутив содержит следующие каталоги:

vfp2c32	Содержит FLL и все вспомогательные файлы.
vfp2c32examples	Содержит пример проекта.
vfp2c32front	VFP2C32 Front — вспомогательная программа, которая транслирует типы C struct, union и enum в код VFP.
vfpsrvhost	VFPSrvHost — это автономный проект на языке C для размещения объектов VFP COM в службе Windows.

Интеграция в ваш проект

Добавьте vfp2c32.fll/vfp2c64.fll и vfp2c.h в свой проект — оба должны быть исключены из приложения/исполняемого файла.

Если вам нужно передать массивы в стиле C в библиотеку C, вам нужно будет взглянуть на классы, определенные в vfp2carray.prg, и также включить их.

vfp2c32t.fll/vfp2c64t.fll — это потокобезопасная версия библиотеки, ее можно использовать в многопоточных библиотеках VFP COM DLL.

Инициализация/очистка библиотеки

```
&& где-то в коде запуска вашей программы
#include vfp2c.h
SET LIBRARY TO vfp2c32.fll ADDITIVE
&& или если используется VFP Advanced 64 бит
SET LIBRARY TO vfp2c64.fll ADDITIVE
```

Развертывание 32 бит

Просто поместите копию vfp2c32t.fll/vfp2c32t.fll и msvc71.dll в структуру каталогов вашего приложения, а **НЕ** в Windows\System32.

Развертывание 64 бит

Поместите копию vfp2c64t.fll/vfp2c64t.fll и msvc100.dll в структуру каталогов вашего приложения, а **НЕ** в Windows\System32.

Intellisense и контекстная помощь

Установка Intellisense

Чтобы установить поддержку Intellisense для функций в VFP2C32, выполните программу «vfp2cinstallintelli.prg» из подкаталога «vfp2c32\help».

Специальные сценарии

Чтобы вывести список всех функций внутри библиотеки, введите «vfp2c» и пробел.

Особенности

Эти функции превышают ограничение IntelliSense в 26 символов:

- AbortRasConnectionNotification
- CreatePublicShadowObjReference
- RasClearConnectionStatistics
- RasConnectionNotificationEx
- RegisterObjectAsFileMoniker
- ReleasePublicShadowObjReference

Когда для этих функций всплывает IntelliSense, отображаются только первые 26 символов. Просто не обращайте внимания и укажите параметры, а когда закончите, вместо ввода последнего символа «)», чтобы закрыть вызов функции, введите еще один «,», после чего скрипт исправит имя функции за вас.

Контекстно-зависимая помощь

Также имеется контекстно-зависимый скрипт справки для функций VFP2C32, который легко интегрируется со справочной системой VFP.

Всякий раз, когда функция, содержащаяся в библиотеке, выделяется в окне редактора, при нажатии клавиши F1 вы сразу переходите к разделу справки внутри этого CHM-файла.

Установка контекстно-зависимой справки

Скрипт контекстной справки зависит от скрипта Intellisense, поэтому сначала установите его, затем скопируйте файлы "vfp2c32_help.prg" и "vfp2c32.chm" из подкаталога "vfp2c32\help" в каталог установки VFP. Скрипт должен выполняться при каждом запуске VFP, для этого добавьте эту строку в скрипт _STARTUP: DO (HOME() + 'vfp2c32_help.prg') WITH 'install' Если у вас еще нет скрипта _STARTUP, вы можете просто скопировать файл "_startup.prg" в каталог установки VFP и выбрать его в "Инструменты" -> "Параметры" -> "Расположение файлов" -> "Автозагрузка программы".

Разработчики и соавторы

Руководитель проекта

Кристиан Эльшайд - c.ehlscheid@gmx.de

Соавторы

- Эрик Селье

Перевод

- Михаил Леменёв

VFP2C32Front

Введение

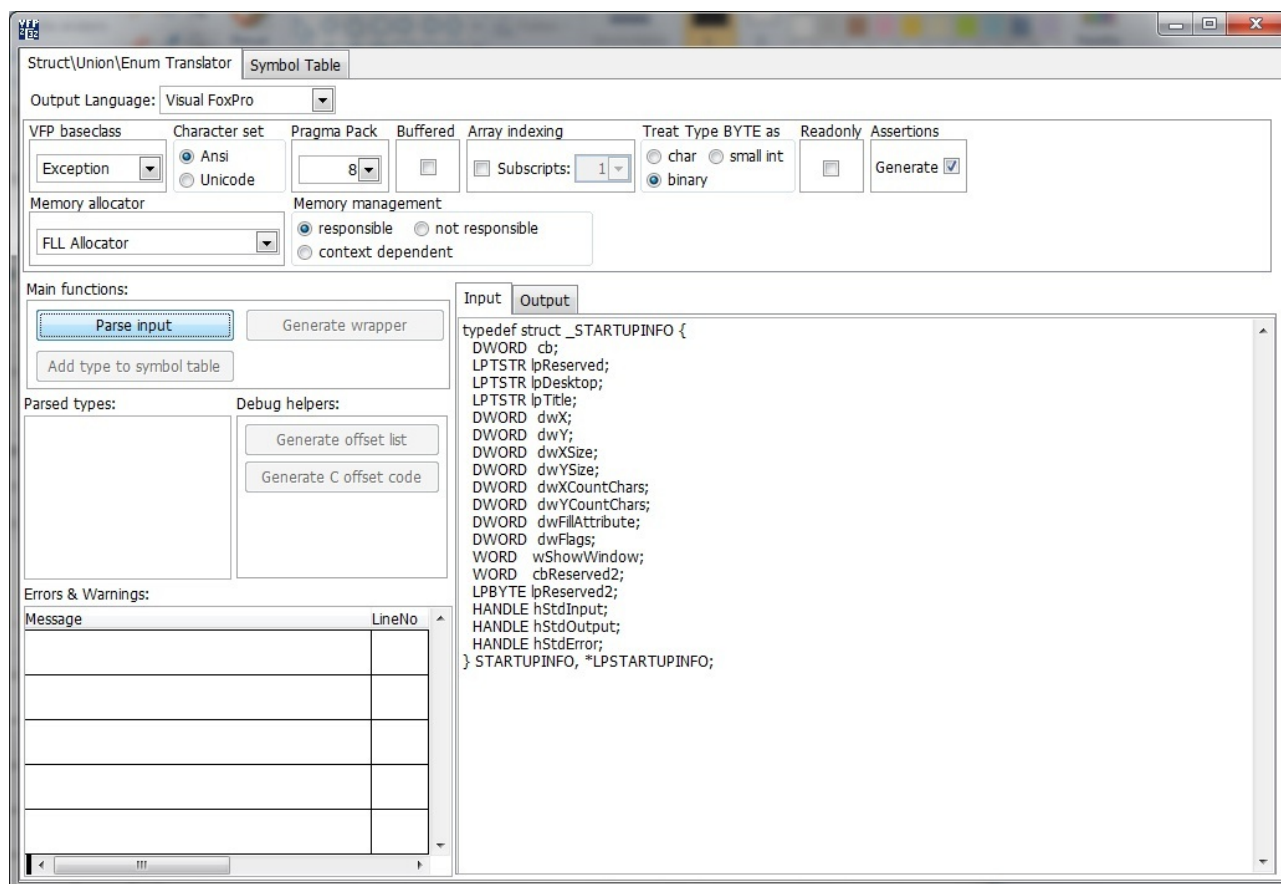
Приложение VFP2C32Front анализирует фрагменты кода C и создает классы VFP, которые эмулируют предоставленные определения типов.

Полученные классы можно использовать как обычные классы VFP, это позволяет программировать в чистом объектно-ориентированном стиле и легко переносить код C в VFP.

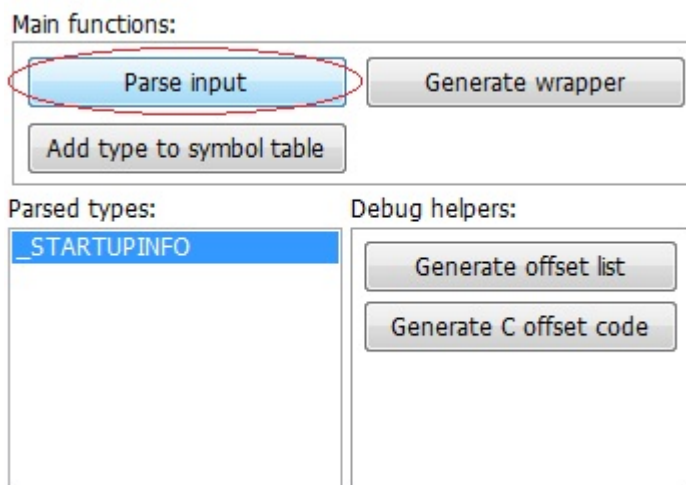
Общее использование

Для преобразования типов C в VFP скопируйте исходный код этих типов в поле редактирования на странице **ввода**.

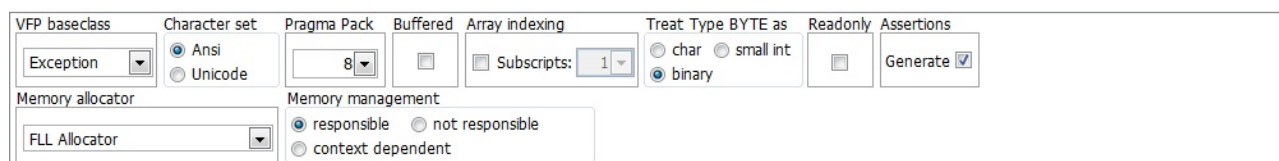
На рисунке 1 показан пример со структурой [STARTUPINFO](#).



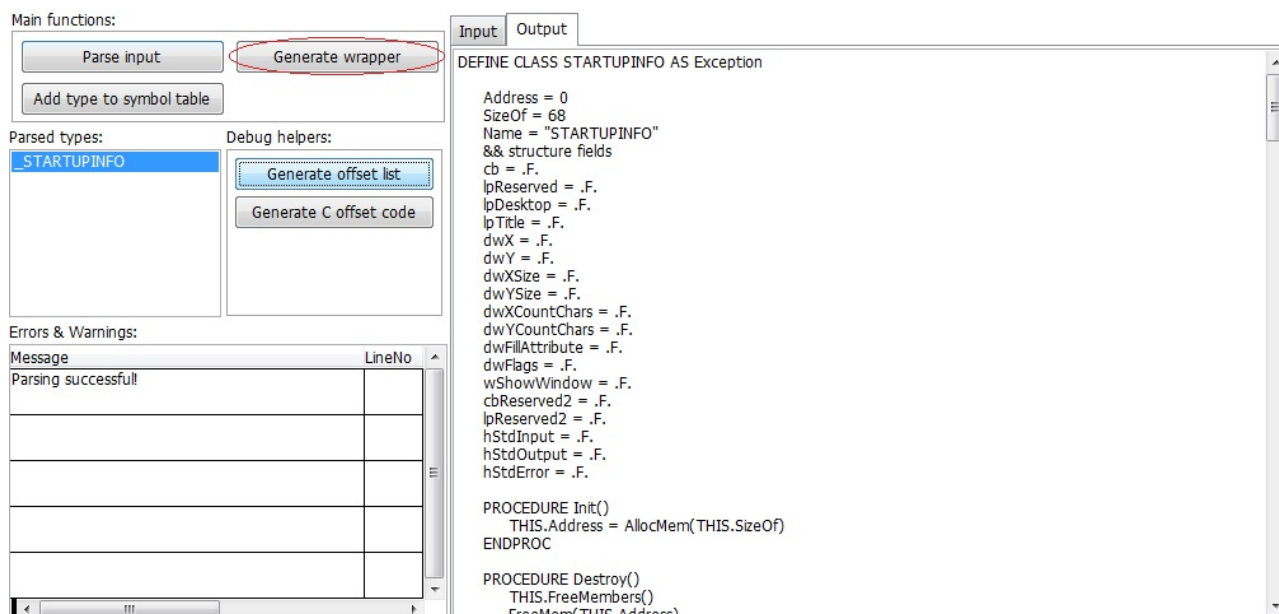
На следующем шаге нажмите кнопку «**Анализ входных данных**». Если все прошло успешно, вы должны увидеть проанализированный тип в списке «**Анализируемые типы**», как показано на рисунке 2.



Теперь вам нужно настроить, как тип должен быть переведен. Параметры, которые управляют тем, как тип будет переведен, расположены вверху, как показано на рисунке 3. Посмотрите тему [Параметры перевода](#), чтобы узнать, как эти параметры влияют на сгенерированный код.



Наконец, нажмите кнопку «**Сгенерировать оболочку**», и сгенерированный код VFP отобразится на странице «**Вывод**», как показано на рисунке 4.



VFP2C32Front - Варианты перевода

VFP baseclass

Указывает базовый класс созданного класса-оболочки, **Exception** или **Relation**. **Exception** — это класс с наименьшими накладными расходами, доступный в VFP 9.

Character set

Для многих функций Windows API на самом деле есть 2 реализации, одна использует набор символов **Ansi**, а другая использует набор символов **Unicode**.

Существует несколько общих типов строк, например LPTSTR, которые сопоставляются либо со строкой Ansi, либо со строкой Unicode, этот параметр определяет, как интерпретируются эти типы.

Pragma pack

Параметр Pragma pack управляет выравниванием упаковки для структур и объединений. Для структур и объединений в API Windows этот параметр редко нуждается в настройке. Просто попробуйте сначала значение по умолчанию. Если у вас возникнут проблемы при передаче структуры в функцию C, вы можете проверить, вычисляется ли правильное смещение для отдельных членов, сравнив вывод **Generate offset list** с выводом скомпилированной программы C, сгенерированной **Generate C offset code**.

Buffered

Эта опция является особым случаем управления памятью для структуры.

Многие функции Windows API ожидают, что вы передадите один большой блок памяти, в который функция затем сохранит массив определенной структуры, включая все данные, на которые ссылаются эти структуры. Отметьте эту опцию, если вы собираетесь использовать такую функцию, например [EnumServicesStatusEx](#), [EnumDependentServices](#).

Эта опция также делает структуру доступной только для чтения.

Array indexing

Эта опция добавляет специальный код, позволяющий легко перебирать массивы структуры. Если вы установите опцию **Buffered**, весьма вероятно, что вам также нужно установить и эту.

Treat type BYTE as

Указывает, как следует читать и записывать любые элементы типа BYTE в структуре.

ReadOnly

Создает только методы доступа для членов структуры или объединения.

Assertions

Если этот флажок установлен, в каждом методе Assign генерируются операторы ASSERT, которые проверяют тип данных и диапазон назначенного значения.

Memory allocator

Указывает, какие функции выделения памяти следует использовать.

Параметр **FLL Allocator** создаст код, который использует пользовательские функции кучи FLL: [AllocMem](#), [ReAllocMem](#), [FreeMem](#).

Параметр **Global Allocator** создаст код, который использует глобальные функции кучи FLL: [AllocHGlobal](#), [ReAllocHGlobal](#), [FreeHGlobal](#), которые являются обертками для функций Windows API [GlobalAlloc](#), [GlobalReAlloc](#) и [GlobalFree](#).

Memory allocator

Этот параметр определяет, как сгенерированный класс-оболочка обрабатывает память. **responsibility** создает код в сгенерированном классе для выделения и освобождения необходимой памяти.

not responsibility не создает никакого кода управления памятью, используйте его для структур, которые используются только как часть других структур.

context dependet создает код для обоих вышеуказанных случаев. Ярким примером этого варианта использования является структура [RECT](#), которая часто используется внутри других структур, но также должна передаваться индивидуально во многих функциях API.

VFP2C32Front - Распространенные ошибки

Ниже приведен список распространенных ошибок, которые могут возникнуть на этапе **ввода данных Parse**.

Ошибка	Решение
синтаксическая ошибка, неожиданный \$end, ожидается ';'!	Каждое определение типа должно заканчиваться символом ';'. Добавьте символ ';' в конце сообщаемого номера строки.
Обнаружен токен препроцессора 'SOMENAME'. Замените его фактическим значением.	Парсер не может разрешить определения препроцессора. Найдите определение внутри заголовочного файла C и замените его фактическим значением. Например, замените MAX_PATH на 261.
Неизвестный спецификатор типа «SOMENAME»!	Сообщаемый тип отсутствует в таблице символов . Сначала его нужно добавить. Эта ошибка часто возникает, если вы переводите структуру, содержащую подструктуры. Сначала переведите подструктуры и используйте функцию Добавить тип в таблицу символов .

VFP2C32Front - GlobalMemoryStatusEx

GlobalMemoryStatusEx

Функция [GlobalMemoryStatusEx](#) ожидает указатель на структуру [MEMORYSTATUSEX](#). Приложение отвечает за выделение и освобождение требуемой памяти для структуры, используются [параметры перевода по умолчанию](#). В созданный код внесено несколько изменений, см. комментарии в примере ниже.

```
LOCAL loMemStatus
m.loMemStatus = CREATEOBJECT('MEMORYSTATUSEX')
? GlobalMemoryStatusEx(m.loMemStatus.Address)
? 'Memory Load', m.loMemStatus.dwMemoryLoad
? 'Physical memory', m.loMemStatus.ullTotalPhys

FUNCTION GlobalMemoryStatusEx(lpBuffer)
  && BOOL WINAPI GlobalMemoryStatusEx( __inout LPMEMORYSTATUSEX lpBuffer)
  DECLARE INTEGER GlobalMemoryStatusEx IN WIN32API INTEGER
  RETURN GlobalMemoryStatusEx(m.lpBuffer)
ENDFUNC

DEFINE CLASS MEMORYSTATUSEX AS Exception

  Address = 0
  SizeOf = 64
```



```
Name = "MEMORYSTATUSEX"
&& structure fields
dwLength = .F.
dwMemoryLoad = .F.
ullTotalPhys = .F.
ullAvailPhys = .F.
ullTotalPageFile = .F.
ullAvailPageFile = .F.
ullTotalVirtual = .F.
ullAvailVirtual = .F.
ullAvailExtendedVirtual = .F.

PROCEDURE Init()
    THIS.Address = AllocMem(THIS.SizeOf)
    THIS.dwLength = THIS.SizeOf && manually added, this member needs to be set
                                && to the size of the structure
ENDPROC

PROCEDURE Destroy()
    FreeMem(THIS.Address)
ENDPROC

PROCEDURE dwLength_Access()
    RETURN ReadUInt(THIS.Address)
ENDPROC

PROCEDURE dwLength_Assign(lnNewVal)
    WriteUInt(THIS.Address, lnNewVal)
ENDPROC

PROCEDURE dwMemoryLoad_Access()
    RETURN ReadUInt(THIS.Address+4)
ENDPROC

PROCEDURE ullTotalPhys_Access()
    RETURN ReadUInt64(THIS.Address+8, 4) && changed to return value
                                         && as numeric value
ENDPROC

PROCEDURE ullAvailPhys_Access()
    RETURN ReadUInt64(THIS.Address+16, 4)
ENDPROC

PROCEDURE ullTotalPageFile_Access()
    RETURN ReadUInt64(THIS.Address+24, 4)
ENDPROC

PROCEDURE ullAvailPageFile_Access()
    RETURN ReadUInt64(THIS.Address+32, 4)
ENDPROC

PROCEDURE ullTotalVirtual_Access()
    RETURN ReadUInt64(THIS.Address+40, 4)
ENDPROC

PROCEDURE ullAvailVirtual_Access()
    RETURN ReadUInt64(THIS.Address+48, 4)
ENDPROC

PROCEDURE ullAvailExtendedVirtual_Access()
    RETURN ReadUInt64(THIS.Address+56, 4)
ENDPROC

ENDDEFINE
```

VFP2C32Front - RegisterPowerSettingNotification

RegisterPowerSettingNotification

Следующий пример отслеживает изменения состояния питания системы.

С помощью функции [RegisterPowerSettingNotification](#) мы регистрируем дескриптор окна, который будет получать уведомления при изменении состояния питания системы.

Затем мы подключаем оконную процедуру окна с помощью функции [BindEventsEx](#).

Структура [POWERBROADCAST_SETTING](#) имеет переменную длину, указатель на эту структуру передается зарегистрированной функции обратного вызова **PowerBroadCastEvent**.

Параметры трансляции для этой структуры: **Memory management - not responsible**, поскольку структура передается нам из ОС, и нам не нужно выделять или освобождать память. Члены

DataDword и **DataGuid** добавляются вручную для доступа к изменяющимся данным в структуре.

```
#DEFINE WM_POWERBROADCAST          0x0218
#DEFINE PBT_POWERSETTINGCHANGE     0x8013

#DEFINE GUID_POWERSCHEME_PERSONALITY 0h41855D2443392244B02513A784F679B7  &&
{245D8541-3943-4422-B025-13A784F679B7}
#DEFINE GUID_MIN_POWER_SAVINGS      0hDA7F5E8CBFE8964A9A85A6E23A8C635C  &&
{8C5E7FDA-E8BF-4A96-9A85-A6E23A8C635C}
#DEFINE GUID_MAX_POWER_SAVINGS      0h081384A14135AB4FBC81F71556F20B4A  &&
{A1841308-3541-4FAB-BC81-F71556F20B4A}
#DEFINE GUID_TYPICAL_POWER_SAVINGS  0h22421B3894F6F0419685FF5BB260DF2E  &&
{381B4222-F694-41F0-9685-FF5BB260DF2E}
#DEFINE GUID_ACDC_POWER_SOURCE      0h599A3E5DD5E9004BA6BDFF34FF516548  &&
{5D3E9A59-E9D5-4B00-A6BD-FF34FF516548}
#DEFINE GUID_BATTERY_PERCENTAGE_REMAINING 0h4180ADA75AB4AE4C87A3EECB468A9E1  &&
{A7AD8041-B45A-4CAE-87A3-EECB468A9E1}
#DEFINE GUID_IDLE_BACKGROUND_TASK   0hD8315C5134F73D16A0FD11A08C91E8F1  &&
{515C31D8-F734-163D-A0FD-11A08C91E8F1}
#DEFINE GUID_SYSTEM_AWAYMODE        0h80F5A798F701AA489C0F44352C29E5C0  &&
{98A7F580-01F7-48AA-9C0F-44352C29E5C0}
#DEFINE GUID_MONITOR_POWER_ON       0h151073021045264599E6E5A17EBD1AEA  &&
{02731015-4510-4526-99E6-E5A17EBD1AEA}

PUBLIC goPowerEvents
m.goPowerEvents = CREATEOBJECT('cPowerEvents')
m.goPowerEvents.RegisterEvent(GUID_POWERSCHEME_PERSONALITY)
m.goPowerEvents.RegisterEvent(GUID_ACDC_POWER_SOURCE)
m.goPowerEvents.RegisterEvent(GUID_BATTERY_PERCENTAGE_REMAINING)
m.goPowerEvents.RegisterEvent(GUID_MONITOR_POWER_ON)

DEFINE CLASS cPowerEvents AS Custom

    PROTECTED oRegisteredEvents
    oRegisteredEvents = .NULL.

    FUNCTION Init
        THIS.oRegisteredEvents = CREATEOBJECT('Collection')
        BINDEVENTSEX( _VFP.hWnd, WM_POWERBROADCAST, THIS, ;
            'PowerBroadCastEvent', 'wParam, lParam')
    ENDFUNC

    FUNCTION Destroy
        THIS.UnregisterEvent()
        UNBINDEVENTSEX( _VFP.hWnd, WM_POWERBROADCAST)
    ENDFUNC

    FUNCTION RegisterEvent(lcEvent)
```

```

LOCAL lnIndex, lnHandle
m.lnIndex = THIS.oRegisteredEvents.GetKey(m.lcEvent)
IF m.lnIndex = 0
    m.lnHandle = RegisterPowerSettingNotification(_VFP.hWnd, m.lcEvent, 0)
    THIS.oRegisteredEvents.Add(m.lnHandle, m.lcEvent)
ENDIF
ENDFUNC

FUNCTION UnregisterEvent(lcEvent)
LOCAL lnIndex, lnHandle
IF VARTYPE(m.lcEvent) = 'C'
    m.lnIndex = THIS.oRegisteredEvents.GetKey(m.lcEvent)
    IF m.lnIndex != 0
        m.lnHandle = THIS.oRegisteredEvents.Item(m.lnIndex)
        UnregisterPowerSettingNotification(m.lnHandle)
        THIS.oRegisteredEvents.Remove(m.lnIndex)
    ENDIF
ELSE && remove all events
    FOR EACH m.lnHandle IN THIS.oRegisteredEvents
        UnregisterPowerSettingNotification(m.lnHandle)
    ENDFOR
    THIS.oRegisteredEvents.Remove(-1)
ENDIF
ENDFUNC

FUNCTION PowerBroadCastEvent
LPARAMETERS dwEvent, lnData

DO CASE

    CASE m.dwEvent = PBT_POWERSETTINGCHANGE

        LOCAL loPBS, lcPowerSetting
        m.loPBS = CREATEOBJECT('POWERBROADCAST_SETTING', m.lnData) && pass
                                                                && pointer to structure!
        m.lcPowerSetting = m.loPBS.PowerSetting.Guid
        DO CASE
            CASE m.lcPowerSetting == GUID_POWERSCHEME_PERSONALITY
                LOCAL lcPSGuid, lcPSName
                m.lcPSGuid = m.loPBS.DataGuid.Guid
                m.lcPSName = ICASE(m.lcPSGuid == GUID_MIN_POWER_SAVINGS, ;
                    'High Performance', ;
                    m.lcPSGuid == GUID_MAX_POWER_SAVINGS, 'Power Saver', 'Automatic')
                ? 'Active power scheme changed to: ' + m.lcPSName

            CASE m.lcPowerSetting == GUID_ACDC_POWER_SOURCE
                LOCAL lnPowerSource
                m.lnPowerSource = m.loPBS.DataDword
                DO CASE
                    CASE m.lnPowerSource = 0
                        ? 'The computer is powered by an AC power source.'
                    CASE m.lnPowerSource = 1
                        ? 'The computer is powered by an onboard battery power source.'
                    CASE m.lnPowerSource = 2
                        ? 'The computer is powered by a short-term power source such'+
                            ' as a UPS device.'
                ENDCASE

            CASE m.lcPowerSetting == GUID_BATTERY_PERCENTAGE_REMAINING
                ? 'Current battery capacity: ' + ;
                    ALLTRIM(STR(m.loPBS.DataDword)) + '%'

            CASE m.lcPowerSetting == GUID_MONITOR_POWER_ON
                ? 'Monitor state changed to: ' + ;

```

```

        IIF(m.loPBS.DataDword = 0, 'Off', 'On')

    ENDCASE

ENDCASE

ENDFUNC

ENDDDEFINE

FUNCTION RegisterPowerSettingNotification(hRecipient, PowerSettingGuid,  Flags)
    DECLARE INTEGER RegisterPowerSettingNotification IN WIN32API ;
        INTEGER hRecipient, STRING PowerSettingGuid, INTEGER Flags
    RETURN RegisterPowerSettingNotification(m.hRecipient, m.PowerSettingGuid, ;
        m.Flags)
ENDFUNC

FUNCTION UnregisterPowerSettingNotification(Handle)
    DECLARE INTEGER UnregisterPowerSettingNotification IN WIN32API INTEGER Handle
    RETURN UnregisterPowerSettingNotification(m.Handle)
ENDFUNC

DEFINE CLASS POWERBROADCAST_SETTING AS Exception

    Address = 0
    Name = "POWERBROADCAST_SETTING"
    && structure fields
    PowerSetting = .NULL.
    DataLength = .F.
    DataGuid = .F. && added manually to access GUID at offset of Data member
    DataDword = .F. && added manually to access DWORD at offset of Data member

    PROCEDURE Init(lnAddress)
        ASSERT TYPE('lnAddress') = 'N' AND lnAddress != 0 ;
        MESSAGE 'Invalid structure address!'
        THIS.Address = lnAddress
        THIS.PowerSetting = CREATEOBJECT('GUID', THIS.Address)
        THIS.DataGuid = CREATEOBJECT('GUID', THIS.Address+20) && added manually
    ENDPROC

    PROCEDURE Address_Assign(lnAddress)
        DO CASE
            CASE THIS.Address = 0
                THIS.Address = lnAddress
            CASE THIS.Address = lnAddress
            OTHERWISE
                THIS.Address = lnAddress
                THIS.PowerSetting.Address = lnAddress
                THIS.DataGuid.Address = lnAddress + 20 && added manually
        ENDCASE
    ENDPROC

    PROCEDURE DataLength_Access()
        RETURN ReadUInt(THIS.Address+16)
    ENDPROC

    PROCEDURE DataDword_Access()
        RETURN ReadUInt(THIS.Address+20)
    ENDPROC

ENDDDEFINE

DEFINE CLASS GUID AS Exception

```

```
Address = 0
SizeOf = 16
PROTECTED Embedded
Embedded = .F.
Name = "GUID"
&& structure fields
Data1 = .F.
Data2 = .F.
Data3 = .F.
Data4 = .F.
Guid = .F. && custom member to assign and access GUID as a binary string
GuidIdString = .F. && custom member to assign and access GUID as a readable
string

PROCEDURE Init(lnAddress)
  IF PCOUNT() = 0
    THIS.Address = AllocMem(THIS.SizeOf)
  ELSE
    THIS.Address = lnAddress
    THIS.Embedded = .T.
  ENDIF
ENDPROC

PROCEDURE Destroy()
  IF !THIS.Embedded
    FreeMem(THIS.Address)
  ENDIF
ENDPROC

PROCEDURE Data1_Access()
  RETURN ReadUInt(THIS.Address)
ENDPROC

PROCEDURE Data1_Assign(lnNewVal)
  WriteUInt(THIS.Address, lnNewVal)
ENDPROC

PROCEDURE Data2_Access()
  RETURN ReadUShort(THIS.Address+4)
ENDPROC

PROCEDURE Data2_Assign(lnNewVal)
  WriteUShort(THIS.Address+4, lnNewVal)
ENDPROC

PROCEDURE Data3_Access()
  RETURN ReadUShort(THIS.Address+6)
ENDPROC

PROCEDURE Data3_Assign(lnNewVal)
  WriteUShort(THIS.Address+6, lnNewVal)
ENDPROC

PROCEDURE Data4_Access()
  RETURN ReadCharArray(THIS.Address+8, 8)
ENDPROC

PROCEDURE Data4_Assign(lnNewVal)
  WriteCharArray(THIS.Address+8, lnNewVal, 8)
ENDPROC

PROCEDURE Guid_Access()
  RETURN ReadBytes(THIS.Address, 16)
ENDPROC
```

```
PROCEDURE Guid_Assign(lnNewVal)
    WriteBytes(THIS.Address, m.lnNewVal, 16)
ENDPROC

PROCEDURE GuidString_Access()
    RETURN STRINGFROMCLSID(THIS.Address)
ENDPROC

PROCEDURE GuidString_Assign(lnNewVal)
    LOCAL lcGuid
    m.lcGuid = CLSIDFROMSTRING(m.lnNewVal)
    WriteBytes(THIS.Address, m.lcGuid)
ENDPROC

ENDDEFINE
```

Функции по категориям

Массивы	Функции, работающие с собственными массивами VFP.
Эмуляция массива C	Функции для преобразования курсоров и массивов VFP в массивы в стиле C и наоборот.
Обратный вызов и события C	Эмуляция функции обратного вызова C и асинхронные события.
Эмуляция структуры C	Функции манипулирования памятью для эмуляции структур C.
COM	Функции, связанные с COM/OLE.
Общие диалоги	Некоторые общие диалоги.
Преобразование	Различные функции преобразования.
Дата и время	Функции, предоставляющие информацию о часовых поясах, преобразуют значения даты и времени в распространенные типы C и обратно.
Файловая система	Извлекайте информацию или манипулируйте файлами и каталогами.
Общая библиотека	Общие библиотечные функции.
Низкоуровневая обработка файлов	Расширенные функции низкоуровневой обработки файлов.
Управление памятью	Распределение памяти.
Разнообразный	Различные полезные функции.
Нетворкинг	Все, что передается по сети.
ODBC	Расширенные функции ODBC.
Информация о принтере	Предоставляет информацию о принтерах.
Информация о процессе	Предоставляет информацию о процессах, запущенных в системе.
RAS	Управление подключениями RAS (удалённого доступа).
Реестр	API для управления реестром.
Информация о ресурсах	Извлечение информации о ресурсах в модулях (dll и исполняемых файлах).
Управление услугами	Получение информации и управление службами Windows.
Оболочка	Обертки вокруг нескольких функций оболочки Windows.
Системная информация	Получение информации о системе.
Информация о томе	Получение информации о томах.
Информация об окне	Получение информации об окнах.

Массивы

Функции, работающие с собственными массивами VFP.

AAverage	Вычисляет среднее арифметическое значений в массиве.
AMax	Извлекает максимальное значение в массиве.
AMin	Извлекает минимальное значение в массиве.
ASum	Вычисляет сумму значений в массиве.

Управление памятью

Распределение памяти.

AllocHGlobal	Выделяет указанное количество байтов из глобальной кучи.
------------------------------	--

AllocMem	Выделяет указанное количество байтов из пользовательской библиотечной кучи.
AllocMemTo	Выделяет указанное количество байтов из пользовательской библиотечной кучи и сохраняет указатель на выделенную память по переданному адресу.
AMemBlocks	Сохраняет информацию обо всех блоках, выделенных из внутренней кучи библиотеки, в массив.
CompactMem	Возвращает размер наибольшего выделенного свободного блока в куче библиотеки.
FreeHGlobal	Освобождает память, выделенную с помощью AllocHGlobal .
FreeMem	Освобождает блок памяти, выделенный из внутренней кучи библиотеки функцией AllocMem или ReAllocMem .
FreePMem	Освобождает блок памяти, на который указывает переданный указатель, из внутренней кучи библиотеки.
FreeRefArray	Освобождает всю память, выделенную для переданного массива в стиле C.
LockHGlobal	Блокирует глобальный объект памяти и возвращает указатель на первый байт блока памяти объекта.
ReAllocHGlobal	Изменяет размер указанного объекта глобальной памяти.
ReAllocMem	Изменяет размер блока памяти, ранее выделенного AllocMem .
SizeOfMem	Возвращает размер блока памяти, выделенного AllocMem .
UnlockHGlobal	Уменьшает количество блокировок, связанных с объектом памяти, выделенным с помощью AllocHGlobal .
ValidateMem	Проверяет внутреннюю кучу библиотеки. Функция сканирует все блоки памяти в куче и проверяет, что структуры управления кучей, поддерживаемые менеджером кучи, находятся в согласованном состоянии. Вы также можете использовать функцию ValidateMem для проверки одного блока памяти без проверки действительности всей кучи.

Реестр

API для управления реестром.

ARegistryKeys	Сохраняет информацию обо всех подразделах указанного раздела реестра в массиве.
ARegistryValues	Сохраняет информацию обо всех значениях указанного раздела реестра в массиве.
CancelRegistryChange	Останавливает поток, отслеживающий изменения указанного ключа реестра.
CloseRegistryKey	Закрывает предоставленный дескриптор ключа реестра (HKEY).
CreateRegistryKey	Создает или открывает раздел реестра.
DeleteRegistryKey	Удаляет указанный раздел реестра.
FindRegistryChange	Отслеживает изменения ключа реестра в отдельном потоке.
OpenRegistryKey	Открывает указанный раздел реестра.
ReadRegistryKey	Извлекает данные для указанного значения реестра.
RegistryHiveToObject	Сохраняет раздел реестра, включая все подразделы, в объекте.
RegistryValuesToObject	Сохраняет все значения ключа реестра в объекте.
WriteRegistryKey	Устанавливает данные и тип указанного значения в разделе реестра.

Нетворкинг

Все, что передается по сети.

AbortUrlDownloadToFileEx	Прерывает асинхронную загрузку, начатую с помощью UrlDownloadToFileEx .
AipAddresses	Сохраняет все IP-адреса машины в массиве.
ANetFiles	Сохраняет информацию об открытых файлах на сервере в массиве.
ANetServers	Сохраняет все серверы указанного типа, видимые в домене, в массиве.
GetServerTime	Извлекает текущее время с сервера Windows.
IcmpPing	Отправляет эхо-запрос IPv4 ICMP, также известный как ping, и возвращает любые эхо-ответы.
Ip2MacAddress	Возвращает MAC-адрес для заданного IP-адреса.
ResolveHostToIp	Определяет IP-адрес для указанного имени хоста.
SyncToSNTPServer	Синхронизирует системное время и дату с SNTP-сервером.
UrlDownloadToFileEx	Загружает ресурсы из Интернета и сохраняет их в файл.

Обратный вызов и события C

Эмуляция функции обратного вызова C и асинхронные события.

AsyncWaitForObject	Ожидает API HANDLE в отдельном потоке без блокировки; при получении сигнала HANDLE выполняется предоставленная функция обратного вызова.
BindEventsEx	Предоставляет возможность выполнить функцию или метод объекта, когда окно API получает указанное сообщение окна.
CancelWaitForObject	Останавливает поток, ожидающий сигнала HANDLE.
CreateCallbackFunc	Создает ассемблерный преобразователь, который эмулирует функцию обратного вызова C.
CreatePublicShadowObjReference	Создает новую публичную переменную, ссылающуюся на предоставленный объект, не увеличивая счетчик ссылок на объекты.
DestroyCallbackFunc	Освобождает переданную функцию обратного вызова C.
ReleasePublicShadowObjReference	Освобождает публичную переменную, ссылающуюся на объект, созданный с помощью CreatePublicShadowObjReference .
UnbindEventsEx	Отменяет привязку событий для окон, которые ранее были привязаны с помощью BindEventsEx .

COM

Функции, связанные с COM/OLE.

CLSIDFromProgID	Извлекает двоичный CLSID для указанного ProgID (имя класса COM), например «VisualFoxPro.Application».
CLSIDFromString	Преобразует понятный человеку CLSID в двоичный CLSID. Это обратная функция StringFromCLSID .
CreateGuid	Создает новый GUID (глобальный уникальный идентификатор).

CreateThreadObject	Создает COM-объект в отдельном потоке.
GetIUnknown	Возвращает указатель на интерфейс IUnknown объекта COM.
IsEqualGuid	Сравнивает два GUID на предмет эквивалентности.
ProgIDFromCLSID	Извлекает ProgID для заданного CLSID.
RegisterActiveObject	Регистрирует переданный объект как активный объект для класса, указанного в cProgID. Регистрация приводит к тому, что объект указывается в таблице запущенных объектов (ROT) OLE, глобально доступной таблице поиска, которая отслеживает объекты, которые в данный момент запущены на компьютере.
RegisterObjectAsFileMoniker	Создает файловое имя для переданного объекта.
RevokeActiveObject	Отменяет регистрацию объекта в таблице текущих объектов (ROT).
StringFromCLSID	Преобразует глобальный уникальный идентификатор (GUID) в строку печатных символов.

Оболочка

Обертки вокруг нескольких функций оболочки Windows.

SHCopyFiles	Копирует один или несколько файлов.
SHDeleteFiles	Удаляет один или несколько файлов.
SHGetShellItem	Возвращает скриптовый прокси-объект через интерфейс IShellItem2.
SHMoveFiles	Перемещает один или несколько файлов.
SHRenameFiles	Переименовывает один или несколько файлов.
SHSpecialFolder	Возвращает путь к специальной папке.

ODBC

Расширенные функции ODBC.

ASQLDataSources	Сохраняет информацию о настроенных источниках данных ODBC в массиве.
ASQLDrivers	Сохраняет информацию об установленных драйверах ODBC в массиве.
ChangeSQLDataSource	Изменяет источник данных ODBC.
CreateSQLDataSource	Создает источник данных ODBC.
DeleteSQLDataSource	Удаляет источник данных ODBC.
SQLCancelEx	Выпускает подготовленный оператор SQL.
SQLColumnPrivilegesEx	Создает курсор столбцов и связанных с ними привилегий для указанной таблицы.
SQLColumnsEx	Создает курсор имен столбцов в указанных таблицах.
SQLCreateTableTypeCursor	Создает пустой курсор указанного зарегистрированного типа таблицы SQL Server. Вы можете заполнить этот курсор, а затем передать ему вызов SQLExecEx для параметров TVP (параметры с табличными значениями).
SQLExecEx	Расширенный SQLEXEC отправляет SQL-оператор в источник данных, где он обрабатывается.
SQLForeignKeysEx	Создает курсор внешних ключей в указанной таблице

	(столбцы в указанной таблице, которые ссылаются на первичные ключи в других таблицах) или курсор внешних ключей в других таблицах, которые ссылаются на первичный ключ в указанной таблице.
SQLGetPropEx	Расширенный SQLGETPROP , извлекает атрибуты соединения ODBC.
SQLPrepareEx	Расширенный SQLPREPARE. Подготавливает SQL-оператор для удаленного выполнения SQLExecEx ().
SQLPrimaryKeysEx	Создает курсор с именами столбцов, составляющих первичный ключ таблицы.
SQLProcedureColumnsEx	Создает курсор входных и выходных параметров, а также столбцов, составляющих результирующий набор для указанных процедур.
SQLProceduresEx	Создает курсор имен процедур, хранящихся в определенном источнике данных.
SQLRegisterTableType	Регистрирует тип таблицы SQL Server в библиотеке для использования в SQLExecEx .
SQLSetPropEx	Расширенный SQLSETPROP , устанавливает атрибуты соединения ODBC.
SQLSpecialColumnsEx	Создает курсор со следующей информацией о столбцах в таблице: - оптимальный набор столбцов, который уникально идентифицирует строку в таблице. - столбцы, которые автоматически обновляются при обновлении любого значения в строке транзакцией.
SQLTablePrivilegesEx	Создает курсор таблиц и привилегий, связанных с каждой таблицей.
SQLTablesEx	Создает курсор с именами таблиц, каталогов или схем, а также типами таблиц, хранящимися в определенном источнике данных.
SQLVariantToValue	Преобразует поле типа SQL_SS_VARIANT, полученное из SQL Server через SQLExecEx , в эквивалентный ему тип FoxPro.
ValueToSQLVariant	Преобразует значение FoxPro в двоичное значение для хранения в двоичное значение SQL_SS_VARIANT для хранения внутри SQL Server.

Управление службами

Получение информации и управление службами Windows.

ADependentServices	Сохраняет все службы, зависящие от переданной службы, в массиве.
AServiceConfig	Сохраняет параметры конфигурации указанной службы Windows в массиве.
AServices	Сохраняет информацию о службах Windows в массиве.
AServiceStatus	Сохраняет информацию о текущем состоянии указанной службы в массиве.
CloseServiceHandleEx	Закрывает предоставленный дескриптор службы (SC_HANDLE).
ContinueService	Отправляет запрос на продолжение указанной службе Windows.
ControlServiceEx	Отправляет пользовательский запрос управления указанной службе Windows.

CreateServiceEx	Устанавливает службу Windows.
OpenServiceEx	Открывает существующую службу.
PauseService	Отправляет запрос на паузу указанной службе.
StartServiceEx	Запускает службу Windows.
StopServiceEx	Отправляет запрос на остановку указанной службе.
WaitForServiceStatus	Отслеживает достижение службой Windows указанного состояния.

RAS

Управление подключениями RAS (удалённого доступа).

AbortRasConnectionNotification	Прерывает поток, отслеживающий соединение RAS, запущенный RasConnectionNotificationEx .
ARasConnections	Сохраняет информацию обо всех активных RAS-подключениях в массиве.
ARasDevices	Сохраняет информацию обо всех доступных устройствах с поддержкой RAS в массиве.
ARasPhonebookEntries	Сохраняет информацию обо всех именах записей в телефонной книге удаленного доступа в массив.
RasClearConnectionStatistics	Очищает всю накопленную статистику для указанного RAS-соединения.
RasConnectionNotificationEx	Запускает новый поток, который отслеживает наличие в системе подключений RAS. При каждом создании или завершении подключения вызывается указанная процедура обратного вызова.
RasDialDlgEx	Устанавливает RAS-соединение, используя указанную запись телефонной книги и учетные данные вошедшего в систему пользователя.
RasDialEx	Функция RasDial устанавливает RAS-соединение между RAS-клиентом и RAS-сервером.
RasGetConnectStatusEx	Извлекает информацию о текущем состоянии указанного подключения удаленного доступа в массив.
RasHangUpEx	Завершает соединение удаленного доступа.
RasPhonebookDlgEx	Отображает главное диалоговое окно Dial-Up Networking. Из этого модального диалогового окна пользователь может набрать номер, изменить или удалить выбранную запись телефонной книги, создать новую запись телефонной книги или указать пользовательские настройки. Функция возвращается, когда диалоговое окно закрывается.

Дата и время

Функции, предоставляющие информацию о часовых поясах, преобразуют значения даты и времени в распространенные типы C и обратно.

ATimeZones	Извлекает информацию о часовых поясах.
Double2DT	Преобразует двойное (числовое) значение в дату и время.
DT2Double	Преобразует значение даты и времени в двойное (числовое) значение.
DT2FT	Преобразует значение datetime в структуру FILETIME .
DT2ST	Преобразует значение datetime в структуру SYSTEMTIME .
DT2Timet	Преобразует значение datetime в метку времени Time_t

	(Unix).
DT2UTC	Преобразует значение даты и времени текущего активного часового пояса в значение даты и времени в формате UTC.
FT2DT	Преобразует структуру FILETIME в значение datetime.
GetSystemTimeEx	Возвращает текущую системную дату и время по всемирному координированному времени (UTC) или по местному времени.
SetSystemTimeEx	Устанавливает текущее системное время и дату.
ST2DT	Преобразует структуру SYSTEMTIME в значение datetime.
Timet2DT	Преобразует временную метку Time_t (Unix) в значение datetime.
UTC2DT	Преобразует значение даты и времени из UTC в местный часовой пояс.

Разное

Различные полезные функции.

AFontInfo	Извлекает информацию о включенных шрифтах из файла шрифтов True Type в объект.
ASplitStr	Разбивает строку на массив на основе указанной длины.
Decimals	Возвращает количество десятичных знаков числового значения.
GetCursorPosEx	Извлекает положение курсора мыши.
Int64_Add	Складывает 64-битные целые числа.
Int64_Div	Делит 64-битные целые числа.
Int64_Mod	Делит одно 64-битное целое число на другое и возвращает остаток.
Int64_Mul	Умножает 64-битные целые числа.
Int64_Sub	Вычитает 64-битные целые числа.

Низкоуровневая обработка файлов

Расширенные функции низкоуровневой обработки файлов.

Основные функции/улучшения.

Также обрабатывает файлы размером более 2 ГБ.

Файлы можно создавать/открывать для общего доступа (собственные функции VFP допускают только монопольный доступ).

Дополнительные функции для блокировки частей файла — FLockFile, FLockFileEx, FUnlockFile и FUnlockFileEx.

FCreateEx и FOpenEx возвращают реальный дескриптор файла Windows, вы можете использовать этот дескриптор для других функций API.

Вы также можете передавать дескрипторы API, не созданные с помощью FCreateEx или FOpenEx, в функции FWriteEx, FPutEx, FReadEx и т. д. ...

AFHandlesEx	Сохраняет все открытые дескрипторы файлов, созданные с помощью FCreateEx или FOpenEx , в массив.
FChSizeEx	Изменяет размер файла, открытого с помощью функции FOpenEx или FCreateEx .
FCloseEx	Очищает и закрывает файл или порт связи, открытый с помощью функции FCreateEx или FOpenEx .
FCreateEx	Создает и открывает файл.

FEOFEx	Определяет, находится ли указатель файла в конце файла.
FFlushEx	Сбрасывает на диск файл, открытый с помощью FCreateEx или FOpenEx .
FGetsEx	Возвращает последовательность байтов из указанного файла или порта связи, открытого с помощью FOpenEx или FCreateEx , до тех пор, пока не встретится возврат каретки.
FLockFile	Блокирует область байтов в файле, открытом с помощью FCreateEx или FOpenEx .
FLockFileEx	Блокирует область байтов в файле, открытом с помощью FCreateEx или FOpenEx .
FOpenEx	Открывает файл.
FputsEx	Записывает строку символов, возврат каретки и перевод строки в файл, открытый с помощью FCreateEx или FOpenEx .
FReadEx	Возвращает указанное количество байтов из файла, открытого с помощью FCreateEx или FOpenEx .
FSeekEx	Перемещает указатель файла в файле, открытом с помощью FCreateEx или FOpenEx .
FUnlockFile	Разблокирует область байтов в файле, открытом с помощью FCreateEx или FOpenEx .
FUnlockFileEx	Разблокирует область байтов в файле, открытом с помощью FCreateEx или FOpenEx .
FWriteEx	Записывает строку символов в файл, открытый с помощью FCreateEx или FOpenEx .

Информация о принтере

Предоставляет информацию о принтерах.

APaperSizes	Сохраняет информацию о поддерживаемых размерах бумаги в массиве.
APrinterForms	Сохраняет информацию о формах, поддерживаемых указанным принтером, в массиве.
APrintersEx	Сохраняет информацию о доступных принтерах, серверах печати, доменах или поставщиках печати в массиве.
APrinterTrays	Сохраняет информацию о доступных лотках для бумаги принтера в массиве.
APrintJobs	Сохраняет информацию об указанном наборе заданий печати для указанного принтера в массиве.

Информация о процессе

Предоставляет информацию о процессах, запущенных в системе.

AHeapBlocks	Сохраняет информацию о блоках кучи, выделенных процессом в массив.
AProcesses	Сохраняет информацию о процессах, запущенных в данный момент в системе, в массиве.
AProcessHeaps	Сохраняет информацию о кучах процесса в массиве.
AProcessModules	Сохраняет информацию о модулях (DLL), загруженных процессом, в массив.
AProcessThreads	Сохраняет информацию о потоках процесса в массиве.

Информация о ресурсах

Извлечение информации о ресурсах в модулях (dll и исполняемых файлах).

AResourceLanguages	Сохраняет информацию о языковых ресурсах указанного типа и имени, связанных с двоичным модулем, в массиве.
AResourceNames	Сохраняет информацию о ресурсах указанного типа в двоичном модуле в массиве.
AResourceTypes	Сохраняет информацию о типах ресурсов внутри двоичного модуля в массиве.

Файловая система

Извлекайте информацию или манипулируйте файлами и каталогами.

ADirectoryInfo	Сохраняет информацию о каталоге в массиве.
ADirEx	Расширенная функция ADIR . Сохраняет информацию о файлах в массиве, курсоре или вызывает функцию обратного вызова для каждого файла.
ADriveInfo	Сохраняет информацию о доступных в данный момент дисках в массиве.
AFileAttributes	Сохраняет атрибуты указанного файла или каталога в массиве.
AFileAttributesEx	Сохраняет расширенные атрибуты указанного файла или каталога в массиве.
CancelFileChange	Останавливает поток, отслеживающий изменения файлов в указанном каталоге.
CompareFileTimes	Сравнивает время последней записи двух файлов.
CopyFileEx	Копирует файл, может быть передана дополнительная функция обратного вызова, которая получает статус во время выполнения операции копирования.
DeleteDirectory	Удаляет каталог, включая все файлы и подкаталоги.
DeleteFileEx	Удаляет файл.
FindFileChange	Отслеживает изменения файлов в каталоге в отдельном потоке.
GetFileAttributesEx2	Извлекает атрибуты файла или каталога.
GetFileOwner	Возвращает владельца файла.
GetFileSizeEx2	Возвращает размер файла.
GetFileTimes	Возвращает время создания, последнего доступа и последней записи файла.
GetLongPathNameEx	Преобразует указанный путь в его длинную форму.
GetShortPathNameEx	Возвращает краткую форму указанного пути.
MoveFileEx	Перемещает файл, может быть передана дополнительная функция обратного вызова, которая получает статус во время выполнения операции перемещения.
SetFileAttributesEx	Изменяет атрибуты указанного файла или набора файлов/каталогов, указанных с помощью подстановочного знака поиска.
SetFileTimes	Устанавливает дату и время создания, последнего доступа или последнего изменения указанного файла или каталога.

Общие диалоги

Некоторые общие диалоги.

GetOpenFileNameEx	Создает диалоговое окно «Открыть», в котором пользователь может указать диск, каталог и имя файла или набора файлов, которые необходимо открыть.
GetSaveFileNameEx	Создает диалоговое окно «Сохранить», в котором пользователь может указать диск, каталог и имя файла для сохранения.
MessageBoxEx	Создает, отображает и управляет окном сообщения. Окно сообщения содержит текст и заголовок сообщения, определяемые приложением, любую иконку и любую комбинацию предопределенных кнопок.
SHBrowseFolder	Отображает диалоговое окно, позволяющее пользователю выбрать папку Shell.

Общая библиотека

Общие библиотечные функции.

AErrorEx	Сохраняет информацию о последней ошибке, произошедшей в библиотеке, в массиве.
FormatMessageEx	Извлекает сообщение об ошибке для указанного номера ошибки.
VFP2CSys	Предоставляет доступ к внутренним ресурсам библиотеки или изменяет глобальное поведение библиотеки.

Эмуляция структуры C

Функции манипулирования памятью для эмуляции структур C.

ReadBytes	Возвращает диапазон байтов по указанному адресу.
ReadChar	Возвращает символ C (отдельный символ) из указанного адреса.
ReadCharArray	Извлекает строку из массива символов в стиле C.
ReadCString	Возвращает строку C из указанного адреса.
ReadDouble	Возвращает число двойной точности (64-битное значение с плавающей точкой) из указанного адреса.
ReadFloat	Возвращает число с плавающей точкой (32-битное значение с плавающей точкой) из указанного адреса.
ReadInt	Извлекает 32-битное целое число из указанного адреса.
ReadInt64	Извлекает 64-битное целое число со знаком из указанного адреса.
ReadInt8	Извлекает 8-битное целое число из указанного адреса.
ReadLogical	Извлекает логическое значение из указанного адреса.
ReadPChar	Возвращает символ C (отдельный символ) из указанного косвенного адреса.
ReadPCString	Возвращает строку C из указанного косвенного адреса.
ReadPDouble	Возвращает число двойной точности (64-битное значение с плавающей точкой) из указанного косвенного адреса.
ReadPFloat	Возвращает число с плавающей точкой (32-битное значение с плавающей точкой) из указанного косвенного адреса.
ReadPInt	Извлекает 32-битное целое число из указанного косвенного

	адреса.
ReadPInt64	Извлекает 64-битное целое число со знаком из указанного косвенного адреса.
ReadPInt8	Извлекает 8-битное целое число из указанного косвенного адреса.
ReadPLogical	Извлекает логическое значение из указанного косвенного адреса.
ReadPointer	Извлекает указатель из указанного адреса.
ReadPPointer	Извлекает указатель из указанного косвенного адреса.
ReadProcessMemoryEx	Извлекает диапазон байтов из области памяти другого процесса.
ReadPShort	Извлекает 16-битное целое число из указанного косвенного адреса.
ReadPUInt	Извлекает 32-битное беззнаковое целое число из указанного косвенного адреса.
ReadPUInt64	Извлекает 64-битное беззнаковое целое число из указанного косвенного адреса.
ReadPUInt8	Извлекает 8-битное беззнаковое целое число из указанного косвенного адреса.
ReadPUSHort	Извлекает 16-битное целое число без знака из указанного косвенного адреса.
ReadPWString	Извлекает строку Unicode, преобразованную в Ansi, из указанного косвенного адреса.
ReadShort	Извлекает 16-битное целое число из указанного адреса.
ReadUInt	Извлекает 32-битное беззнаковое целое число из указанного адреса.
ReadUInt64	Извлекает 64-битное беззнаковое целое число из указанного адреса.
ReadUInt8	Извлекает 8-битное беззнаковое целое число из указанного адреса.
ReadUShort	Извлекает 16-битное беззнаковое целое число из указанного адреса.
ReadWCharArray	Извлекает строку из массива символов Unicode в стиле C.
ReadWString	Извлекает строку Unicode, преобразованную в Ansi, из указанного адреса.
WriteBytes	Записывает двоичные данные по указанному адресу.
WriteChar	Записывает один символ по указанному адресу.
WriteCharArray	Записывает строку по указанному адресу.
WriteCString	Выделяет или перераспределяет строку в стиле C.
WriteDouble	Записывает число двойной точности (64-битное с плавающей точкой) по указанному адресу.
WriteFloat	Записывает число с плавающей точкой (32-битное число с плавающей точкой) по указанному адресу.
WriteGPCString	Выделяет или перевыделяет память для строки в стиле C и записывает указатель на эту строку по указанному адресу.
WriteInt	Записывает 32-битное целое число по указанному адресу.
WriteInt64	Записывает 64-битное целое число со знаком по указанному адресу.
WriteInt8	Записывает 8-битное целое число по указанному адресу.
WriteLogical	Записывает логическое значение по указанному адресу.

WritePChar	Записывает один символ по указанному косвенному адресу.
WritePCString	Выделяет или перевыделяет память для строки в стиле С и записывает указатель на эту строку по указанному адресу.
WritePDouble	Записывает число двойной точности (64-битное с плавающей точкой) по указанному косвенному адресу.
WritePFloat	Записывает число с плавающей точкой (32-битное число с плавающей точкой) по указанному косвенному адресу.
WritePInt	Записывает 32-битное целое число по указанному косвенному адресу.
WritePInt64	Записывает 64-битное целое число со знаком по указанному косвенному адресу.
WritePInt8	Записывает 8-битное целое число по указанному косвенному адресу.
WritePLogical	Записывает логическое значение по указанному косвенному адресу.
WritePointer	Записывает указатель по указанному адресу.
WritePPointer	Записывает указатель по указанному косвенному адресу.
WritePShort	Записывает 16-битное целое число по указанному косвенному адресу.
WritePUInt	Записывает 32-битное беззнаковое целое число по указанному косвенному адресу.
WritePUInt64	Записывает 64-битное беззнаковое целое число по указанному косвенному адресу.
WritePUInt8	Записывает 8-битное беззнаковое целое число по указанному косвенному адресу.
WritePUShort	Записывает 16-битное беззнаковое целое число по указанному косвенному адресу.
WritePWChar	Записывает один символ Unicode по указанному косвенному адресу.
WritePWString	Выделяет или перераспределяет память для строки Unicode в стиле С и записывает указатель на эту строку по указанному адресу.
WriteShort	Записывает 16-битное целое число по указанному адресу.
WriteUInt	Записывает 32-битное беззнаковое целое число по указанному адресу.
WriteUInt64	Записывает 64-битное беззнаковое целое число по указанному адресу.
WriteUInt8	Записывает 8-битное беззнаковое целое число по указанному адресу.
WriteUShort	Записывает 16-битное беззнаковое целое число по указанному адресу.
WriteWChar	Записывает один символ Unicode по указанному адресу.
WriteWCharArray	Записывает строку Unicode по указанному адресу.
WriteWString	Выделяет или перераспределяет строку Unicode в стиле С.

Эмуляция массива C

Функции для преобразования курсоров и массивов VFP в массивы в стиле С и наоборот.

MarshalCArray2Cursor	Преобразует массив С в курсор VFP.
MarshalCArray2FoxArray	Преобразует массив С в массив VFP.
MarshalCursor2CArray	Преобразует курсор VFP в массив С.

[MarshalFoxArray2CArray](#) Преобразует массив VFP в массив C.

Преобразование

Различные функции преобразования.

Colors2RGB	Преобразует предоставленные значения красного, зеленого и синего в 32-битное целое число.
Double2Str	Преобразует двойное (64-битное числовое) значение в двоичную строку.
Float2Str	Преобразует значение с плавающей точкой (32-битное числовое значение) в двоичную строку.
Int642Str	Преобразует значение <code>__int64</code> (64-битное числовое значение) в требуемый формат.
Long2Str	Преобразует длинное знаковое числовое значение (32-битное число) в двоичную строку.
Num2Binary	Преобразует 32-битное целое число в двоичную строку, удобочитаемую человеком.
PG_ByteA2Str	Преобразует значение PostgreSQL ByteA в строку.
PG_Str2ByteA	Преобразует строку в экранированное значение PostgreSQL ByteA.
RGB2Colors	Разбивает RGB на составные части.
Short2Str	Преобразует короткое знаковое значение (16-битное числовое значение) в двоичную строку.
Str2Double	Преобразует двоичную строку в число двойной точности (64-битное числовое значение).
Str2Float	Преобразует двоичную строку в число с плавающей точкой (32-битное числовое значение).
Str2Int64	Преобразует двоичную строку в <code>__int64</code> (64-битное числовое значение).
Str2Long	Преобразует двоичную строку в длинное число со знаком (32-битное числовое значение).
Str2Short	Преобразует двоичную строку в короткое число со знаком (16-битное числовое значение).
Str2UInt64	Преобразует двоичную строку в беззнаковое <code>__int64</code> (64-битное числовое значение).
Str2ULong	Преобразует двоичную строку в беззнаковое длинное число (32-битное числовое значение).
Str2UShort	Преобразует двоичную строку в беззнаковое короткое число (16-битное числовое значение).
UInt642Str	Преобразует беззнаковое значение <code>__int64</code> (64-битное числовое значение) в требуемый формат.
ULong2Str	Преобразует беззнаковое длинное (32-битное числовое значение) значение в двоичную строку.
UShort2Str	Преобразует беззнаковое короткое (16-битное числовое значение) значение в двоичную строку.
Value2Variant	Преобразует переменную или поле любого типа в двоичную строку.
Variant2Value	Преобразует двоичную строку, созданную Value2Variant , в исходное значение.

Информация о томе

Получение информации о томах.

AVolumeInformation	Сохраняет информацию о файловой системе и томе, связанных с указанным корневым каталогом, в массиве.
AVolumeMountPoints	Сохраняет имена смонтированных папок на указанном томе в массиве.
AVolumePaths	Сохраняет буквы дисков и пути GUID томов для указанного тома в массиве.
AVolumes	Сохраняет имена томов на компьютере в массиве.

Системная информация

Получение информации о системе.

ADesktopArea	Сохраняет размер видимой области рабочего стола (не включая системную панель задач или панели инструментов рабочего стола приложений) в массиве.
ADesktops	Сохраняет все рабочие столы, связанные с указанной оконной станцией вызывающего процесса, в массив.
Устройства отображения	Сохраняет информацию об устройствах отображения в текущем сеансе в массив.
AResolutions	Сохраняет информацию обо всех графических режимах (разрешениях) для устройства отображения в массиве.
AWindowStations	Сохраняет имена всех оконных станций в текущем сеансе в массив.
ExpandEnvironmentStringsEx	Расширяет переменные среды в переданной строке.
GetLocaleInfoEx	Извлекает информацию о локали.
GetSystemDirectoryEx	Возвращает путь к системному каталогу Windows.
GetWindowsDirectoryEx	Возвращает путь к каталогу Windows.
OsEx	Возвращает текущую операционную систему.

Информация об окне

Получение информации об окнах.

AWindowProps	Сохраняет информацию о записях в списке свойств окна в массиве.
AWindows	Сохраняет дескрипторы окон (HWND) в массиве.
AWindowsEx	Сохраняет информацию об окнах в массиве.
CenterWindowEx	Центрирует окно либо в определенном окне, либо в его родительском окне, либо на рабочем столе.
GetWindowRectEx	Возвращает размеры ограничивающего прямоугольника указанного окна.
GetWindowTextEx	Извлекает текст, соответствующий окну.

Функции по алфавиту

А

AAverage	Вычисляет среднее арифметическое значений в массиве.
AbortRasConnectionNotification	Прерывает поток, отслеживающий соединение RAS, запущенный RasConnectionNotificationEx .
AbortUrlDownloadToFileEx	Прерывает асинхронную загрузку, начатую с помощью UrlDownloadToFileEx .
ADependentServices	Сохраняет все службы, зависящие от переданной службы, в массиве.
ADesktopArea	Сохраняет размер видимой области рабочего стола (не включая системную панель задач или панели инструментов рабочего стола приложений) в массиве.
ADesktops	Сохраняет все рабочие столы, связанные с указанной оконной станцией вызывающего процесса, в массив.
ADirectoryInfo	Сохраняет информацию о каталоге в массиве.
ADirEx	Расширенная функция ADIR . Сохраняет информацию о файлах в массиве, курсоре или вызывает функцию обратного вызова для каждого файла.
ADisplayDevices	Сохраняет информацию об устройствах отображения в текущем сеансе в массив.
ADriveInfo	Сохраняет информацию о доступных в данный момент дисках в массиве.
AErrorEx	Сохраняет информацию о последней ошибке, произошедшей в библиотеке, в массиве.
AFHandlesEx	Сохраняет все открытые дескрипторы файлов, созданные с помощью FCreateEx или FOpenEx , в массив.
AFileAttributes	Сохраняет атрибуты указанного файла или каталога в массиве.
AFileAttributesEx	Сохраняет расширенные атрибуты указанного файла или каталога в массиве.
AFontInfo	Извлекает информацию о включенных шрифтах из файла шрифтов True Type в объект.
AHeapBlocks	Сохраняет информацию о блоках кучи, выделенных процессом в массив.
AIpAddresses	Сохраняет все IP-адреса машины в массиве.
AllocHGlobal	Выделяет указанное количество байтов из глобальной кучи.
AllocMem	Выделяет указанное количество байтов из пользовательской библиотечной кучи.
AllocMemTo	Выделяет указанное количество байтов из пользовательской библиотечной кучи и сохраняет указатель на выделенную память по переданному адресу.
AMax	Извлекает максимальное значение в массиве.
AMemBlocks	Сохраняет информацию обо всех блоках, выделенных из внутренней кучи библиотеки, в массив.
AMin	Извлекает минимальное значение в массиве.
ANetFiles	Сохраняет информацию об открытых файлах на сервере в массиве.
ANetServers	Сохраняет все серверы указанного типа, видимые в домене, в массиве.
APaperSizes	Сохраняет информацию о поддерживаемых размерах бумаги в

	массиве.
APrinterForms	Сохраняет информацию о формах, поддерживаемых указанным принтером, в массиве.
APrintersEx	Сохраняет информацию о доступных принтерах, серверах печати, доменах или поставщиках печати в массиве.
APrinterTrays	Сохраняет информацию о доступных лотках для бумаги принтера в массиве.
APrintJobs	Сохраняет информацию об указанном наборе заданий печати для указанного принтера в массиве.
AProcesses	Сохраняет информацию о процессах, запущенных в данный момент в системе, в массиве.
AProcessHeaps	Сохраняет информацию о кучах процесса в массиве.
AProcessModules	Сохраняет информацию о модулях (DLL), загруженных процессом, в массиве.
AProcessThreads	Сохраняет информацию о потоках процесса в массиве.
ARasConnections	Сохраняет информацию обо всех активных RAS-подключениях в массиве.
ARasDevices	Сохраняет информацию обо всех доступных устройствах с поддержкой RAS в массиве.
ARasPhonebookEntries	Сохраняет информацию обо всех именах записей в телефонной книге удаленного доступа в массиве.
ARegistryKeys	Сохраняет информацию обо всех подразделах указанного раздела реестра в массиве.
ARegistryValues	Сохраняет информацию обо всех значениях указанного раздела реестра в массиве.
AResolutions	Сохраняет информацию обо всех графических режимах (разрешениях) для устройства отображения в массиве.
AResourceLanguages	Сохраняет информацию о языковых ресурсах указанного типа и имени, связанных с двоичным модулем, в массиве.
AResourceNames	Сохраняет информацию о ресурсах указанного типа в двоичном модуле в массиве.
AResourceTypes	Сохраняет информацию о типах ресурсов внутри двоичного модуля в массиве.
AServiceConfig	Сохраняет параметры конфигурации указанной службы Windows в массиве.
AServices	Сохраняет информацию о службах Windows в массиве.
AServiceStatus	Сохраняет информацию о текущем состоянии указанной службы в массиве.
ASplitStr	Разбивает строку на массив на основе указанной длины.
ASQLDataSources	Сохраняет информацию о настроенных источниках данных ODBC в массиве.
ASQLDrivers	Сохраняет информацию об установленных драйверах ODBC в массиве.
ASum	Вычисляет сумму значений в массиве.
AsyncWaitForObject	Ожидает API HANDLE в отдельном потоке без блокировки; при получении сигнала HANDLE выполняется предоставленная функция обратного вызова.
ATimeZones	Извлекает информацию о часовых поясах.
AVolumeInformation	Сохраняет информацию о файловой системе и томе, связанных с указанным корневым каталогом, в массиве.
AVolumeMountPoints	Сохраняет имена смонтированных папок на указанном томе в массиве.

AVolumePaths	Сохраняет буквы дисков и пути GUID томов для указанного тома в массиве.
AVolumes	Сохраняет имена томов на компьютере в массиве.
AWindowProps	Сохраняет информацию о записях в списке свойств окна в массиве.
AWindows	Сохраняет дескрипторы окон (HWND) в массиве.
AWindowsEx	Сохраняет информацию об окнах в массиве.
AWindowStations	Сохраняет имена всех оконных станций в текущем сеансе в массив.

B

BindEventsEx	Предоставляет возможность выполнить функцию или метод объекта, когда окно API получает указанное сообщение окна.
------------------------------	--

C

CancelFileChange	Останавливает поток, отслеживающий изменения файлов в указанном каталоге.
CancelRegistryChange	Останавливает поток, отслеживающий изменения указанного ключа реестра.
CancelWaitForObject	Останавливает поток, ожидающий сигнала HANDLE.
CenterWindowEx	Центрирует окно либо в определенном окне, либо в его родительском окне, либо на рабочем столе.
ChangeSQLDataSource	Изменяет источник данных ODBC.
CloseRegistryKey	Закрывает предоставленный дескриптор ключа реестра (HKEY).
CloseServiceHandleEx	Закрывает предоставленный дескриптор службы (SC_HANDLE).
CLSIDFromProgID	Извлекает двоичный CLSID для указанного ProgID (имя класса COM), например «VisualFoxPro.Application».
CLSIDFromString	Преобразует понятный человеку CLSID в двоичный CLSID. Это обратная функция StringFromCLSID .
Colors2RGB	Преобразует предоставленные значения красного, зеленого и синего в 32-битное целое число.
CompactMem	Возвращает размер наибольшего выделенного свободного блока в куче библиотеки.
CompareFileTimes	Сравнивает время последней записи двух файлов.
ContinueService	Отправляет запрос на продолжение указанной службе Windows.
ControlServiceEx	Отправляет пользовательский запрос управления указанной службе Windows.
CopyFileEx	Копирует файл, может быть передана дополнительная функция обратного вызова, которая получает статус во время выполнения операции копирования.
CreateCallbackFunc	Создает ассемблерный преобразователь, который эмулирует функцию обратного вызова C.
CreateGuid	Создает новый GUID (глобальный уникальный идентификатор).
CreatePublicShadowObjReference	Создает новую публичную переменную, ссылающуюся на предоставленный объект, не увеличивая счетчик ссылок на объекты.
CreateRegistryKey	Создает или открывает раздел реестра.
CreateServiceEx	Устанавливает службу Windows.

CreateSQLDataSource	Создает источник данных ODBC.
CreateThreadObject	Создает COM-объект в отдельном потоке.

D

Decimals	Возвращает количество десятичных знаков числового значения.
DeleteDirectory	Удаляет каталог, включая все файлы и подкаталоги.
DeleteFileEx	Удаляет файл.
DeleteRegistryKey	Удаляет указанный раздел реестра.
DeleteSQLDataSource	Удаляет источник данных ODBC.
DestroyCallbackFunc	Освобождает переданную функцию обратного вызова C.
Double2DT	Преобразует двойное (числовое) значение в дату и время.
Double2Str	Преобразует двойное (64-битное числовое) значение в двоичную строку.
DT2Double	Преобразует значение даты и времени в двойное (числовое) значение.
DT2FT	Преобразует значение datetime в структуру FILETIME .
DT2ST	Преобразует значение datetime в структуру SYSTEMTIME .
DT2Timet	Преобразует значение datetime в метку времени Time_t (Unix).
DT2UTC	Преобразует значение даты и времени текущего активного часового пояса в значение даты и времени в формате UTC.

E

ExpandEnvironmentStringsEx	Расширяет переменные среды в переданной строке.
--	---

F

FChSizeEx	Изменяет размер файла, открытого с помощью функции FOpenEx или FCreateEx .
FCloseEx	Очищает и закрывает файл или порт связи, открытый с помощью функции FCreateEx или FOpenEx .
FCreateEx	Создает и открывает файл.
FEofEx	Определяет, находится ли указатель файла в конце файла.
FFlushEx	Сбрасывает на диск файл, открытый с помощью FCreateEx или FOpenEx .
FGetsEx	Возвращает последовательность байтов из указанного файла или порта связи, открытого с помощью FOpenEx или FCreateEx , до тех пор, пока не встретится возврат каретки.
FindFileChange	Отслеживает изменения файлов в каталоге в отдельном потоке.
FindFileChangeEx	Отслеживает изменения файлов в каталоге в отдельном потоке.
FindRegistryChange	Отслеживает изменения ключа реестра в отдельном потоке.
Float2Str	Преобразует значение с плавающей точкой (32-битное числовое значение) в двоичную строку.
FLockFile	Блокирует область байтов в файле, открытом с помощью FCreateEx или FOpenEx .
FLockFileEx	Блокирует область байтов в файле, открытом с помощью FCreateEx или FOpenEx .
FOpenEx	Открывает файл.
FormatMessageEx	Извлекает сообщение об ошибке для указанного номера

	ошибки.
FPutEx	Записывает строку символов, возврат каретки и перевод строки в файл, открытый с помощью FCreateEx или FOpenEx .
FReadEx	Возвращает указанное количество байтов из файла, открытого с помощью FCreateEx или FOpenEx .
FreeHGlobal	Освобождает память, выделенную с помощью AllocHGlobal .
FreeMem	Освобождает блок памяти, выделенный из внутренней кучи библиотеки функцией AllocMem или ReAllocMem .
FreePMem	Освобождает блок памяти, на который указывает переданный указатель, из внутренней кучи библиотеки.
FreeRefArray	Освобождает всю память, выделенную для переданного массива в стиле C.
FSeekEx	Перемещает указатель файла в файле, открытом с помощью FCreateEx или FOpenEx .
FT2DT	Преобразует структуру FILETIME в значение datetime.
FUnlockFile	Разблокирует область байтов в файле, открытом с помощью FCreateEx или FOpenEx .
FUnlockFileEx	Разблокирует область байтов в файле, открытом с помощью FCreateEx или FOpenEx .
FWriteEx	Записывает строку символов в файл, открытый с помощью FCreateEx или FOpenEx .

G

GetCursorPosEx	Извлекает положение курсора мыши.
GetFileAttributesEx2	Извлекает атрибуты файла или каталога.
GetFileOwner	Возвращает владельца файла.
GetFileSizeEx2	Возвращает размер файла.
GetFileTimes	Возвращает время создания, последнего доступа и последней записи файла.
GetIUnknown	Возвращает указатель на интерфейс IUnknown объекта COM.
GetLocaleInfoEx	Извлекает информацию о локали.
GetLongPathNameEx	Преобразует указанный путь в его длинную форму.
GetOpenFileNameEx	Создает диалоговое окно «Открыть», в котором пользователь может указать диск, каталог и имя файла или набора файлов, которые необходимо открыть.
GetSaveFileNameEx	Создает диалоговое окно «Сохранить», в котором пользователь может указать диск, каталог и имя файла для сохранения.
GetServerTime	Извлекает текущее время с сервера Windows.
GetShortPathNameEx	Возвращает краткую форму указанного пути.
GetSystemDirectoryEx	Возвращает путь к системному каталогу Windows.
GetSystemTimeEx	Возвращает текущую системную дату и время по всемирному координированному времени (UTC) или по местному времени.
GetWindowRectEx	Возвращает размеры ограничивающего прямоугольника указанного окна.
GetWindowsDirectoryEx	Возвращает путь к каталогу Windows.
GetWindowTextEx	Извлекает текст, соответствующий окну.

I

IcmpPing	Отправляет эхо-запрос IPv4 ICMP, также известный как ping, и возвращает любые эхо-ответы.
--------------------------	---

Int642Str	Преобразует значение __int64 (64-битное числовое значение) в требуемый формат.
Int64_Add	Складывает 64-битные целые числа.
Int64_Div	Делит 64-битные целые числа.
Int64_Mod	Делит одно 64-битное целое число на другое и возвращает остаток.
Int64_Mul	Умножает 64-битные целые числа.
Int64_Sub	Вычитает 64-битные целые числа.
Ip2MacAddress	Возвращает MAC-адрес для заданного IP-адреса.
IsEqualGuid	Сравнивает два GUID на предмет эквивалентности.

L

LockHGlobal	Блокирует глобальный объект памяти и возвращает указатель на первый байт блока памяти объекта.
Long2Str	Преобразует длинное знаковое числовое значение (32-битное число) в двоичную строку.

M

MarshalCArray2Cursor	Преобразует массив C в курсор VFP.
MarshalCArray2FoxArray	Преобразует массив C в массив VFP.
MarshalCursor2CArray	Преобразует курсор VFP в массив C.
MarshalFoxArray2CArray	Преобразует массив VFP в массив C.
MessageBoxEx	Создает, отображает и управляет окном сообщения. Окно сообщения содержит текст и заголовок сообщения, определяемые приложением, любую иконку и любую комбинацию предопределенных кнопок.
MoveFileEx	Перемещает файл, может быть передана дополнительная функция обратного вызова, которая получает статус во время выполнения операции перемещения.

N

Num2Binary	Преобразует 32-битное целое число в двоичную строку, удобочитаемую человеком.
----------------------------	---

O

OpenRegistryKey	Открывает указанный раздел реестра.
OpenServiceEx	Открывает существующую услугу.
OsEx	Возвращает текущую операционную систему.

P

PauseService	Отправляет запрос на паузу указанной службе.
PG_ByteA2Str	Преобразует значение PostgreSQL ByteA в строку.
PG_Str2ByteA	Преобразует строку в экранированное значение PostgreSQL ByteA.
ProgIDFromCLSID	Извлекает ProgID для заданного CLSID.

R

RasClearConnectionStatistics	Очищает всю накопленную статистику для указанного RAS-соединения.
RasConnectionNotificationEx	Запускает новый поток, который отслеживает наличие в системе подключений RAS. При каждом создании или завершении подключения вызывается указанная процедура обратного вызова.

RasDialDlgEx	Устанавливает RAS-соединение, используя указанную запись телефонной книги и учетные данные вошедшего в систему пользователя.
RasDialEx	Функция RasDial устанавливает RAS-соединение между RAS-клиентом и RAS-сервером.
RasGetConnectStatusEx	Извлекает информацию о текущем состоянии указанного подключения удаленного доступа в массив.
RasHangUpEx	Завершает соединение удаленного доступа.
RasPhonebookDlgEx	Отображает главное диалоговое окно Dial-Up Networking. Из этого модального диалогового окна пользователь может набрать номер, изменить или удалить выбранную запись телефонной книги, создать новую запись телефонной книги или указать пользовательские настройки. Функция возвращается, когда диалоговое окно закрывается.
ReadBytes	Возвращает диапазон байтов по указанному адресу.
ReadChar	Возвращает символ С (отдельный символ) из указанного адреса.
ReadCharArray	Извлекает строку из массива символов в стиле С.
ReadCString	Возвращает строку С из указанного адреса.
ReadDouble	Возвращает число двойной точности (64-битное значение с плавающей точкой) из указанного адреса.
ReadFloat	Возвращает число с плавающей точкой (32-битное значение с плавающей точкой) из указанного адреса.
ReadInt	Извлекает 32-битное целое число из указанного адреса.
ReadInt64	Извлекает 64-битное целое число со знаком из указанного адреса.
ReadInt8	Извлекает 8-битное целое число из указанного адреса.
ReadLogical	Извлекает логическое значение из указанного адреса.
ReadPChar	Возвращает символ С (отдельный символ) из указанного косвенного адреса.
ReadPCString	Возвращает строку С из указанного косвенного адреса.
ReadPDouble	Возвращает число двойной точности (64-битное значение с плавающей точкой) из указанного косвенного адреса.
ReadPFloat	Возвращает число с плавающей точкой (32-битное значение с плавающей точкой) из указанного косвенного адреса.
ReadPInt	Извлекает 32-битное целое число из указанного косвенного адреса.
ReadPInt64	Извлекает 64-битное целое число со знаком из указанного косвенного адреса.
ReadPInt8	Извлекает 8-битное целое число из указанного косвенного адреса.
ReadPLogical	Извлекает логическое значение из указанного косвенного адреса.
ReadPointer	Извлекает указатель из указанного адреса.
ReadPPointer	Извлекает указатель из указанного косвенного адреса.
ReadProcessMemoryEx	Извлекает диапазон байтов из области памяти другого процесса.
ReadPShort	Извлекает 16-битное целое число из указанного косвенного адреса.
ReadPUInt	Извлекает 32-битное беззнаковое целое число из указанного косвенного адреса.

ReadPUInt64	Извлекает 64-битное беззнаковое целое число из указанного косвенного адреса.
ReadPUInt8	Извлекает 8-битное беззнаковое целое число из указанного косвенного адреса.
ReadPUShort	Извлекает 16-битное целое число без знака из указанного косвенного адреса.
ReadPWString	Извлекает строку Unicode, преобразованную в Ansi, из указанного косвенного адреса.
ReadRegistryKey	Извлекает данные для указанного значения реестра.
ReadShort	Извлекает 16-битное целое число из указанного адреса.
ReadUInt	Извлекает 32-битное беззнаковое целое число из указанного адреса.
ReadUInt64	Извлекает 64-битное беззнаковое целое число из указанного адреса.
ReadUInt8	Извлекает 8-битное беззнаковое целое число из указанного адреса.
ReadUShort	Извлекает 16-битное целое число без знака из указанного адреса.
ReadWCharArray	Извлекает строку из массива символов Unicode в стиле C.
ReadWString	Извлекает строку Unicode, преобразованную в Ansi, из указанного адреса.
ReAllocHGlobal	Изменяет размер указанного объекта глобальной памяти.
ReAllocMem	Изменяет размер блока памяти, ранее выделенного AllocMem .
RegisterActiveObject	Регистрирует переданный объект как активный объект для класса, указанного в cProgID. Регистрация приводит к тому, что объект указывается в таблице запущенных объектов (ROT) OLE, глобально доступной таблице поиска, которая отслеживает объекты, которые в данный момент запущены на компьютере.
RegisterObjectAsFileMoniker	Создает файловое имя для переданного объекта.
RegistryHiveToObject	Сохраняет раздел реестра, включая все подразделы, в объекте.
RegistryValuesToObject	Сохраняет все значения ключа реестра в объекте.
ReleasePublicShadowObjReference	Освобождает публичную переменную, ссылающуюся на объект, созданный с помощью CreatePublicShadowObjReference .
ResolveHostToIp	Определяет IP-адрес для указанного имени хоста.
RevokeActiveObject	Отменяет регистрацию объекта в таблице текущих объектов (ROT).
RGB2Colors	Разбивает RGB на составные части.

S

SetFileAttributesEx	Изменяет атрибуты указанного файла или набора файлов/каталогов, указанных с помощью подстановочного знака поиска.
SetFileTimes	Устанавливает дату и время создания, последнего доступа или последнего изменения указанного файла или каталога.
SetSystemTimeEx	Устанавливает текущее системное время и дату.
SHBrowseFolder	Отображает диалоговое окно, позволяющее пользователю выбрать папку Shell.
SHCopyFiles	Копирует один или несколько файлов.

SHDeleteFiles	Удаляет один или несколько файлов.
SHGetShellItem	Возвращает скриптовый прокси-объект через интерфейс IShellItem2.
SHMoveFiles	Перемещает один или несколько файлов.
Short2Str	Преобразует короткое знаковое значение (16-битное числовое значение) в двоичную строку.
SHRenameFiles	Переименовывает один или несколько файлов.
SHSpecialFolder	Возвращает путь к специальной папке.
SizeOfMem	Возвращает размер блока памяти, выделенного AllocMem .
SQLCancelEx	Выпускает подготовленный оператор SQL.
SQLColumnPrivilegesEx	Создает курсор столбцов и связанных с ними привилегий для указанной таблицы.
SQLColumnsEx	Создает курсор имен столбцов в указанных таблицах.
SQLCreateTableTypeCursor	Создает пустой курсор указанного зарегистрированного типа таблицы SQL Server. Вы можете заполнить этот курсор, а затем передать ему вызов SQLExecEx для параметров TVP (параметры с табличными значениями).
SQLExecEx	Расширенный SQLEXEC отправляет SQL-оператор в источник данных, где он обрабатывается.
SQLForeignKeysEx	Создает курсор внешних ключей в указанной таблице (столбцы в указанной таблице, которые ссылаются на первичные ключи в других таблицах) или курсор внешних ключей в других таблицах, которые ссылаются на первичный ключ в указанной таблице.
SQLGetPropEx	Расширенный SQLGETPROP , извлекает атрибуты соединения ODBC.
SQLGetTypeInfoEx	
SQLPrepareEx	Расширенный SQLPREPARE. Подготавливает SQL-оператор для удаленного выполнения SQLExecEx ().
SQLPrimaryKeysEx	Создает курсор с именами столбцов, составляющих первичный ключ таблицы.
SQLProcedureColumnsEx	Создает курсор входных и выходных параметров, а также столбцов, составляющих результирующий набор для указанных процедур.
SQLProceduresEx	Создает курсор имен процедур, хранящихся в определенном источнике данных.
SQLRegisterTableType	Регистрирует тип таблицы SQL Server в библиотеке для использования в SQLExecEx .
SQLSetPropEx	Расширенный SQLSETPROP , устанавливает атрибуты соединения ODBC.
SQLSpecialColumnsEx	Создает курсор со следующей информацией о столбцах в таблице: - оптимальный набор столбцов, который уникально идентифицирует строку в таблице. - столбцы, которые автоматически обновляются при обновлении любого значения в строке транзакцией.
SQLStatisticsEx	
SQLTablePrivilegesEx	Создает курсор таблиц и привилегий, связанных с каждой таблицей.
SQLTablesEx	Создает курсор с именами таблиц, каталогов или схем, а также типами таблиц, хранящимися в определенном источнике данных.
SQLVariantToValue	Преобразует поле типа SQL_SS_VARIANT, полученное из SQL

	Server через SQLExecEx , в эквивалентный ему тип FoxPro.
ST2DT	Преобразует структуру SYSTEMTIME в значение datetime.
StartServiceEx	Запускает службу Windows.
StopServiceEx	Отправляет запрос на остановку указанной службе.
Str2Double	Преобразует двоичную строку в число двойной точности (64-битное числовое значение).
Str2Float	Преобразует двоичную строку в число с плавающей точкой (32-битное числовое значение).
Str2Int64	Преобразует двоичную строку в __int64 (64-битное числовое значение).
Str2Long	Преобразует двоичную строку в длинное число со знаком (32-битное числовое значение).
Str2Short	Преобразует двоичную строку в короткое число со знаком (16-битное числовое значение).
Str2UInt64	Преобразует двоичную строку в беззнаковое __int64 (64-битное числовое значение).
Str2ULong	Преобразует двоичную строку в беззнаковое длинное число (32-битное числовое значение).
Str2UShort	Преобразует двоичную строку в беззнаковое короткое число (16-битное числовое значение).
StringFromCLSID	Преобразует глобальный уникальный идентификатор (GUID) в строку печатных символов.
SyncToSNTPServer	Синхронизирует системное время и дату с SNTP-сервером.

T

Timet2DT	Преобразует временную метку Time_t (Unix) в значение datetime.
--------------------------	--

U

UInt642Str	Преобразует беззнаковое значение __int64 (64-битное числовое значение) в требуемый формат.
ULong2Str	Преобразует беззнаковое длинное (32-битное числовое значение) значение в двоичную строку.
UnbindEventsEx	Отменяет привязку событий для окон, которые ранее были привязаны с помощью BindEventsEx .
UnlockHGlobal	Уменьшает количество блокировок, связанных с объектом памяти, выделенным с помощью AllocHGlobal .
UrlDownloadToFileEx	Загружает ресурсы из Интернета и сохраняет их в файл.
UShort2Str	Преобразует беззнаковое короткое (16-битное числовое значение) значение в двоичную строку.
UTC2DT	Преобразует значение даты и времени из UTC в местный часовой пояс.

V

ValidateMem	Проверяет внутреннюю кучу библиотеки. Функция сканирует все блоки памяти в куче и проверяет, что структуры управления кучей, поддерживаемые менеджером кучи, находятся в согласованном состоянии. Вы также можете использовать функцию ValidateMem для проверки одного блока памяти без проверки действительности всей кучи.
Value2Variant	Преобразует переменную или поле любого типа в двоичную строку.
ValueToSQLVariant	Преобразует значение FoxPro в двоичное значение для

	хранения в двоичное значение SQL_SS_VARIANT для хранения внутри SQL Server.
Variant2Value	Преобразует двоичную строку, созданную Value2Variant , в исходное значение.
VFP2C32	Фиктивная функция для проверки того, загружена ли библиотека.
VFP2CSys	Предоставляет доступ к внутренним ресурсам библиотеки или изменяет глобальное поведение библиотеки.

W

WaitForServiceStatus	Отслеживает достижение службой Windows указанного состояния.
WriteBytes	Записывает двоичные данные по указанному адресу.
WriteChar	Записывает один символ по указанному адресу.
WriteCharArray	Записывает строку по указанному адресу.
WriteCString	Выделяет или перераспределяет строку в стиле C.
WriteDouble	Записывает число двойной точности (64-битное с плавающей точкой) по указанному адресу.
WriteFloat	Записывает число с плавающей точкой (32-битное число с плавающей точкой) по указанному адресу.
WriteGPCString	Выделяет или перевыделяет память для строки в стиле C и записывает указатель на эту строку по указанному адресу.
WriteInt	Записывает 32-битное целое число по указанному адресу.
WriteInt64	Записывает 64-битное целое число со знаком по указанному адресу.
WriteInt8	Записывает 8-битное целое число по указанному адресу.
WriteLogical	Записывает логическое значение по указанному адресу.
WritePChar	Записывает один символ по указанному косвенному адресу.
WritePCString	Выделяет или перевыделяет память для строки в стиле C и записывает указатель на эту строку по указанному адресу.
WritePDouble	Записывает число двойной точности (64-битное с плавающей точкой) по указанному косвенному адресу.
WritePFloat	Записывает число с плавающей точкой (32-битное число с плавающей точкой) по указанному косвенному адресу.
WritePint	Записывает 32-битное целое число по указанному косвенному адресу.
WritePint64	Записывает 64-битное целое число со знаком по указанному косвенному адресу.
WritePint8	Записывает 8-битное целое число по указанному косвенному адресу.
WritePLogical	Записывает логическое значение по указанному косвенному адресу.
WritePointer	Записывает указатель по указанному адресу.
WritePPointer	Записывает указатель по указанному косвенному адресу.
WritePShort	Записывает 16-битное целое число по указанному косвенному адресу.
WritePUInt	Записывает 32-битное беззнаковое целое число по указанному косвенному адресу.
WritePUInt64	Записывает 64-битное беззнаковое целое число по указанному косвенному адресу.
WritePUInt8	Записывает 8-битное беззнаковое целое число по указанному

	косвенному адресу.
WritePUShort	Записывает 16-битное беззнаковое целое число по указанному косвенному адресу.
WritePWChar	Записывает один символ Unicode по указанному косвенному адресу.
WritePWString	Выделяет или перераспределяет память для строки Unicode в стиле C и записывает указатель на эту строку по указанному адресу.
WriteRegistryKey	Устанавливает данные и тип указанного значения в разделе реестра.
WriteShort	Записывает 16-битное целое число по указанному адресу.
WriteUInt	Записывает 32-битное беззнаковое целое число по указанному адресу.
WriteUInt64	Записывает 64-битное беззнаковое целое число по указанному адресу.
WriteUInt8	Записывает 8-битное беззнаковое целое число по указанному адресу.
WriteUShort	Записывает 16-битное беззнаковое целое число по указанному адресу.
WriteWChar	Записывает один символ Unicode по указанному адресу.
WriteWCharArray	Записывает строку Unicode по указанному адресу.
WriteWString	Выделяет или перераспределяет строку Unicode в стиле C.

AAverage

Вычисляет среднее арифметическое значений в массиве.

```
AAverage ( @aArray [, nDimension ])
```

Параметры

@aArray

Массив по ссылке, над которым должно быть выполнено вычисление.

nDimension (необязательно)

по умолчанию = 1

Номер колонки массива, для которой должно быть выполнено вычисление. Если передать 0 в этот параметр, вычисление будет выполнено для всех элементов массива.

Возвращаемое значение

Среднее значение значений, содержащихся в переданном массиве, или .NULL., если все строки были .NULL.

Замечания

Допустимые типы значений в массиве: числовые (N) или денежные (Y). Все значения в массиве, над которыми выполняется вычисление, должны быть одного типа, в противном случае возникнет ошибка «недопустимые аргументы».

Пример

```
LOCAL laTest[20], xj
FOR xj = 1 TO 20
    laTest[xj] = xj
ENDFOR

? AAverage(@laTest) && Возвращает 10.5
```

Смотрите также

Ссылки

[AMax](#)

[AMin](#)

[ASum](#)

AbortRasConnectionNotification

Прерывает поток, отслеживающий соединение RAS, запущенный [RasConnectionNotificationEx](#).

```
AbortRasConnectionNotification( nThreadHandle )
```

Параметры

nThreadHandle

Дескриптор потока, возвращенный из [RasConnectionNotificationEx](#).

Возвращаемое значение

.Т. если поток, отслеживающий соединение RAS, был прерван, .F. в противном случае.

Смотрите также

Ссылки

[ARasConnections](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)

[RasConnectionNotificationEx](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasGetConnectStatusEx](#)

[RasHangUpEx](#)

[RasPhonebookDlgEx](#)

Используемые функции WinApi

[SetEvent](#)

AbortUrlDownloadToFileEx

Прерывает асинхронную загрузку, начатую с помощью [UrlDownloadToFileEx](#).

```
AbortUrlDownloadToFileEx( nThreadHandle )
```

Параметры

nThreadHandle

Дескриптор потока, возвращенный [UrlDownloadToFileEx](#).

Возвращаемое значение

.T. если поток, выполняющий UrlDownloadToFile, был остановлен, .F. в противном случае.

Смотрите также

Ссылки

[AIPAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[ICMPing](#)

[Ip2MacAddress](#)

[Resolve_HostToIp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

ADependentServices

Сохраняет все службы, зависящие от переданной службы, в массиве.

```
ADependentServices( cArrayName , cServiceName | nServiceHandle [,  
cServer [, cDatabase ]])
```

Параметры

cArrayName

Имя массива в виде строки, в которую функция должна сохранить информацию.

По возвращении массив содержит следующую информацию из структур [ENUM_SERVICE_STATUS](#) и [SERVICE_STATUS](#).

Столбец	Описание
1	Имя услуги
2	Отображаемое имя
3	Тип обслуживания
4	Текущее состояние
5	Win32ExitCode
6	ServiceSpecificExitCode
7	CheckPoint
8	ControlsAccepted

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

cServer (необязательно)

Имя сервера, на котором запущена служба.
См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.
См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

Количество зависимых услуг.

Смотрите также

Ссылки

[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ПродолжитьService](#)
[ControlServiceEx](#)
[СоздатьServiceEx](#)
[ОткрытьServiceEx](#)
[ПаузаService](#)
[StartServiceEx](#)
[ОстановитьServiceEx](#)
[ОжиданиеServiceStatus](#)

Используемые функции WinApi

[EnumDependentServices](#)[OpenSCManager](#)[OpenService](#)[CloseServiceHandle](#)

ADesktopArea

Сохраняет размер видимой области рабочего стола (не включая системную панель задач или панели инструментов рабочего стола приложений) в массиве.

```
ADesktopArea( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующие значения из структуры [RECT](#).

Элемент	Значение
1	Left
2	Right
3	Top
4	Bottom

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADesktops](#)
[ADisplayDevices](#)
[AResolutions](#)
[AWindowStations](#)
[ExpandEnvironmentStringsEx](#)
[GetLocaleInfoEx](#)
[GetSystemDirectoryEx](#)
[GetWindowsDirectoryEx](#)
[OsEx](#)

Используемые функции WinApi

[SystemParametersInfo](#) с параметром SPI_GETWORKAREA

ADesktops

Сохраняет все рабочие столы, связанные с указанной оконной станцией вызывающего процесса, в массиве.

```
ADesktops( cArrayName [, nHWINSTA ] )
```

Параметры

cArrayName

Столбец	Значение
1	Имя рабочего стола

nHWINSTA (необязательно)

по умолчанию = windowsstation, возвращаемая [GetProcessWindowStation](#).

Дескриптор API HWINSTA.

Возвращаемое значение

Количество рабочих столов.

Смотрите также

Ссылки

[ADesktopArea](#)
[ADisplayDevices](#)
[AResolutions](#)
[AWindowStations](#)
[ExpandEnvironmentStringsEx](#)
[GetLocaleInfoEx](#)
[GetSystemDirectoryEx](#)
[GetWindowsDirectoryEx](#)
[OsEx](#)

Используемые функции WinApi

[EnumDesktops](#)
[GetProcessWindowStation](#)

ADirectoryInfo

Сохраняет информацию о каталоге в массиве.

```
ADirectoryInfo( cArrayName , cDirectory )
```

Параметры

cArrayname

Имя массива, в котором должна храниться информация о каталоге.

Элемент	Значение
1	Количество файлов внутри каталога, включая файлы внутри подкаталогов
2	Количество подкаталогов
3	Размер каталога - сумма размеров всех файлов

cDirectory

Полный путь к каталогу.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[FindFirstFile](#)
[FindNextFile](#)
[FindClose](#)

ADirEx

Расширенная функция [ADIR](#). Сохраняет информацию о файлах в массиве, курсоре или вызывает функцию обратного вызова для каждого файла.

```
ADirEx( cArrayName | cCursorName | cCallback , cFileSkeleton [,
ncFileFilter [, nFlags [, nMaxRecursion , cFields [,
nFilterMatch ]]]])
```

Параметры

cArrayName | cCursorName | cCallback

Имя массива, курсора или функции обратного вызова.

Структура массива

Столбец	Значение	Тип данных
1	Имя файла	C
2	Короткое имя файла	C
3	Время создания	T
4	Время последнего доступа	T
5	Время последней записи	T
6	Размер файла	N
7	Атрибуты	N или C

Если в параметре nFlags указано ADIREX_DEST_CURSOR и курсор не существует, будет создан курсор с именами полей по умолчанию.

```
CREATE CURSOR theCursorName (filename M, dosfilename C(13),
creationtime T, accesstime T, writetime T, filesize N(20,0),
fileattribs I)
&& или если передан ADIREX_STRING_FILEATTRIBUTES
CREATE CURSOR theCursorName (filename M, dosfilename C(13),
creationtime T, accesstime T, writetime T, filesize N(20,0),
fileattribs V(10))
```

Также возможен самосозданный курсор с дополнительными полями.

Если курсор уже существует, новые записи будут добавлены для каждого найденного файла.

Если вы указываете ADIREX_DEST_CALLBACK в параметре nFlags, определение функции процедуры обратного вызова должно соответствовать этому:

```
FUNCTION
AdirExCallbackExample(cFileName,cDosFileName,tCreationTime,tLastAccessTime,tLastWriteTime,nFileSize,ncFileAttributes)
ENDFUNC
```

cFileSkeleton

Полная строка поиска (диск:\путь\подстановочный знак), например "C:\Winnt*.dll"

ncFileFilter (добавочный)

Дополнительно фильтруйте файлы по атрибутам, тип фильтрации контролируется параметром nFlags .

0 или " или пропущен -> фильтр не используется.

Комбинация числовых атрибутов файла или строка, где каждый символ представляет атрибут. При передаче строки знак '-' перед атрибутами исключает файлы с определенным набором атрибутов, того же самого можно добиться при передаче числового значения, объединив этот параметр с параметром "nFilterMatch".

Атрибут	Символ
FILE_ATTRIBUTE_READONLY	R
FILE_ATTRIBUTE_HIDDEN	H
FILE_ATTRIBUTE_SYSTEM	S
FILE_ATTRIBUTE_DIRECTORY	D
FILE_ATTRIBUTE_ARCHIVE	A
FILE_ATTRIBUTE_TEMPORARY	T
FILE_ATTRIBUTE_SPARSE_FILE	P
FILE_ATTRIBUTE_REPARSE_POINT	L
FILE_ATTRIBUTE_COMPRESSED	C
FILE_ATTRIBUTE_OFFLINE	O
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	I
FILE_ATTRIBUTE_ENCRYPTED	E
FILE_ATTRIBUTE_FAKEDIRECTORY	K

Если указать пользовательский флаг FILE_ATTRIBUTE_FAKEDIRECTORY, то также будут перечислены «поддельные» каталоги «.» и «..».

nFlags (добавочный)

Параметр *nFlags* управляет тремя поведением функции:

1. указывает значение параметра 1

ADIREX_DEST_ARRAY - файлы сохраняются в переданном массиве name

ADIREX_DEST_CURSOR - файлы сохраняются в переданном курсоре name

ADIREX_DEST_CALLBACK - переданная функция name вызывается для каждого найденного файла

2. поведение применяемой фильтрации:

по умолчанию по крайней мере один из указанных атрибутов, переданных в *nFileFilter*, должен быть установлен для совпадения:

ADIREX_FILTER_ALL - все атрибуты должны быть установлены для совпадения

ADIREX_FILTER_NONE - ни один из атрибутов не должен быть установлен для совпадения

ADIREX_FILTER_EXACT - значение, возвращаемое [GetFileAttributes](#), должно быть точно таким же, как предоставленный фильтр

3. преобразование времени файла:

по умолчанию время файла преобразуется в местный часовой пояс, но если указать ADIREX.UTC_TIMES, время файла возвращается в формате времени UTC (GMT).

4. различные функции:

ADIREX_RECURSE - рекурсия в подкаталоги.

ADIREX_FULLPATH - возврат полного пути к файлам

ADIREX_DISABLE_FSREDIRECTION - отключение перенаправления файловой системы для 32/64-битных системных каталогов

ADIREX_STRING_FILEATTRIBUTES - атрибуты файла будут возвращены в строковой форме вместо числового значения

nMaxRecursion: по умолчанию 0 (неограничено)

Максимальная глубина подкаталога для поиска при указании ADIREX_RECURSE.

cFields: по умолчанию все поля в порядке по умолчанию

Список полей, разделенных запятыми, для возврата.

Порядок полей сохраняется для курсоров и массивов, а также для функций обратного вызова.

Должно быть одним из "filename, dosfilename, creationtime, accesstime, writetime, filesize, fileattribs, cfileattribs".

nFilterMatch (добавочный)

При передаче файлы фильтруются по атрибутам, переданным в «ncFileFilter» и этом параметре, по следующей логике.

```
BITAND(fileattributes, ncFileFilter) == nFilterMatch
```

Возвращаемое значение

Количество файлов/каталогов.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[FindFirstFile](#)
[FindNextFile](#)
[FindClose](#)

ADisplayDevices

Сохраняет информацию об устройствах отображения в текущем сеансе в массиве.

```
ADisplayDevices( cArrayName [, cDeviceName ])
```

Параметры

cArrayName

Имя массива в виде строки, в которую функция должна сохранить информацию.

По возвращении массив содержит следующие значения из структуры [DISPLAY_DEVICE](#).

Столбец	Значение	Тип данных
1	DeviceString	C
2	DeviceName	C
3	DeviceID	C
4	DeviceKey	C
5	StateFlags	N

cDeviceName (необязательно)

по умолчанию = NULL

Имя устройства. Если NULL, функция возвращает информацию об адаптере(ах) дисплея на машине.

Для получения дополнительной информации см. Примечания.

Возвращаемое значение

Количество устройств отображения.

Замечания

Ознакомьтесь с разделом примечаний к API [EnumDisplayDevices](#) о том, как получить информацию для мониторов и т. д.

Смотрите также

Ссылки

[ADesktopArea](#)
[ADesktops](#)
[AResolutions](#)
[AWindowStations](#)
[ExpandEnvironmentStringsEx](#)
[GetLocaleInfoEx](#)
[GetSystemDirectoryEx](#)
[GetWindowsDirectoryEx](#)
[OsEx](#)

Используемые функции WinApi

[EnumDisplayDevices](#)

ADriveInfo

Сохраняет информацию о доступных в данный момент дисках в массиве.

```
ADriveInfo( cArrayName )
```

Параметры

cArrayName

Имя массива в виде строки, в который функция должна сохранить информацию.

Столбец	Значение
1	Буква диска
2	Тип диска (возвращается из GetDriveType .)
3	Номер устройства или -1, если DeviceIoControl не удалось
4	Номер раздела или -1, если DeviceIoControl не удалось

Возвращаемое значение

Количество дисков.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[GetLogicalDrives](#)
[GetDriveType](#)
[DeviceIoControl](#)
[CreateFile](#)
[CloseHandle](#)

AErrorEx

Сохраняет информацию о последней ошибке, произошедшей в библиотеке, в массиве.

```
AErrorEx( cArrayName )
```

Параметры

cArrayname

Имя массива, в котором сохраняется дополнительная информация об ошибках последнего вызова FLL:

Примечание

Если ошибок не произошло, AERROREX() не создает массив.

Структура массива.

Столбец	Значение
1	номер ошибки последнего вызова API (GetLastError())
2	имя функции API, которая сообщила об ошибке
3	сообщение об ошибке
4	Значение состояния ODBC (устанавливается только для вызовов, связанных с ODBC)

Возвращаемое значение

Количество ошибок.

Смотрите также

Ссылки

[FormatMessageEx](#)
[VFP2CSys](#)

AFHandlesEx

Сохраняет все открытые дескрипторы файлов, созданные с помощью [FCreateEx](#) или [FOpenEx](#), в массиве.

```
AFHandlesEx( cArrayName )
```

Параметры

cArrayName

Имя массива, в который функция должна сохранять дескрипторы файлов.

Структура массива:

Столбец	Значение
1	HANDLE для файла, коммуникационного порта, канала и т. д.

Возвращаемое значение

Количество дескрипторов файлов.

Пример

```
&& дескрипторы открытых в данный момент файлов  
LOCAL laFileHandles[1]  
? AFHandlesEx('laFileHandles')
```

Смотрите также

Ссылки

[FChSizeEx](#)
[FCloseEx](#)
[FCreateEx](#)
[FEofEx](#)
[FFlushEx](#)
[FGetEx](#)
[FLockFile](#)
[FLockFileEx](#)
[FOpenEx](#)
[FPutsEx](#)
[FReadEx](#)
[FSeekEx](#)
[FUnlockFile](#)
[FUnlockFileEx](#)
[FWriteEx](#)

AFileAttributes

Сохраняет атрибуты указанного файла или каталога в массиве.

```
AFileAttributes( cArrayName , cFileName [, bUTCTimes [,  
bStringAttributes ]])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [WIN32_FILE_ATTRIBUTE_DATA](#).

Элемент	Значение	Тип данных
1	Атрибуты	N или C
2	Размер файла	N
3	Время создания	T
4	Время последнего доступа	T
5	Время последней записи	T

cFileName

Имя файла, для которого необходимо получить атрибуты.

bUTCTimes (необязательно)

по умолчанию = .F.

Если .T., то время файлов извлекается как время UTC, в противном случае оно преобразуется в текущий местный часовой пояс.

bStringAttributes (необязательно)

по умолчанию = .F.

Если .T. атрибуты файла извлекаются в строковой форме.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)

[SetFileAttributesEx](#)

[SetFileTimes](#)

Используемые функции WinApi

[GetFileAttributesEx](#)

AFileAttributesEx

Сохраняет расширенные атрибуты указанного файла или каталога в массиве.

```
AFileAttributesEx( cArrayName , cFileName [, bUTCTimes [,
bStringAttributes ]])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [BY_HANDLE_FILE_INFORMATION](#).

Элемент	Значение	Тип данных
1	Атрибуты	N или C
2	Размер файла	N
3	Время создания	T
4	Время последнего доступа	T
5	Время последней записи	T
6	Количество ссылок на этот файл. Для файловой системы FAT этот член всегда равен 0. Для файловой системы NTFS он может быть больше 1.	N
7	Серийный номер тома, содержащего файл.	N
8	Младшая часть уникального идентификатора, связанного с файлом. Идентификатор (младшая и старшая части) и серийный номер тома однозначно идентифицируют файл на одном компьютере. Чтобы определить, представляют ли два открытых дескриптора один и тот же файл, объедините идентификатор и серийный номер тома для каждого файла и сравните их.	N
9	Старшая часть уникального идентификатора, связанного с файлом.	N

cFileName

Имя файла, для которого необходимо получить атрибуты.

bUTCTimes (необязательно)

по умолчанию = .F.

Если .T., то время файлов извлекается как время UTC, в противном случае оно преобразуется в текущий местный часовой пояс.

bStringAttributes (необязательно)

по умолчанию = .F.

Если .T. атрибуты файла извлекаются в строковой форме.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)
[AFileAttributes](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[GetFileInformationByHandle](#)
[CreateFile](#)
[CloseHandle](#)

AFontInfo

Извлекает информацию о включенных шрифтах из файла шрифтов True Type в объект.

```
AFontInfo( cFileName [, nLanguageId [, nPlatformId ]])
```

Параметры

cFileName

Полное имя файла TTF (шрифт True Type).

nLanguageId (необязательно)

по умолчанию = [GetSystemDefaultLangID\(\)](#)

язык, для которого необходимо получить информацию.

nPlatformId (необязательно)

по умолчанию = PLATFORMID_WINDOWS

Возвращаемое значение

Объект со свойствами, содержащими информацию о файле шрифта.

Смотрите также

Ссылки

[ASplitStr](#)
[Decimals](#)
[GetCursorPosEx](#)
[Int64_Add](#)
[Int64_Div](#)
[Int64_Mod](#)
[Int64_Mul](#)
[Int64_Sub](#)

Используемые функции WinApi

[GetSystemDefaultLangID](#)

AHeapBlocks

Сохраняет информацию о блоках кучи, выделенных процессом в массив.

```
AHeapBlocks( cArrayName , nProcessID , nHeapID )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [HEAPENTRY32](#).

Столбец	Значение	Тип данных
1	Дескриптор блока кучи.	N
2	Линейный адрес начала блока.	N
3	Размер блока кучи в байтах.	N
4	Флаги	N

nProcessID

Идентификатор контекста процесса, которому принадлежит куча.

nHeapID

Идентификатор кучи, подлежащей перечислению.

Возвращаемое значение

Количество блоков кучи.

Смотрите также

Ссылки

[AProcesses](#)
[AProcessHeaps](#)
[AProcessModules](#)
[AProcessThreads](#)

Используемые функции WinApi

[Heap32First](#)
[Heap32Next](#)

AIpAddresses

Сохраняет все IP-адреса машины в массиве.

```
AIpAddresses ( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение
1	IP-адрес

Возвращаемое значение

Количество IP-адресов.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[ICMPing](#)

[Ip2MacAddress](#)

[Resolve HostToIp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

Используемые функции WinApi

[gethostname](#)

[gethostbyname](#)

AllocHGlobal

Выделяет указанное количество байтов из глобальной кучи.

```
AllocHGlobal( nNumberOfBytes [, nFlags ])
```

Параметры

nNumberOfBytes

Количество выделяемых байтов.

nFlags (необязательный, дополнительный)

по умолчанию = GMEM_MOVEABLE | GMEM_ZEROINIT

Одно или комбинация следующих значений.

Флаг	Описание
GMEM_FIXED	Выделяет фиксированную память. Возвращаемое значение — указатель.
GMEM_MOVEABLE	Выделяет перемещаемую память. Блоки памяти никогда не перемещаются в физической памяти, но их можно перемещать в пределах кучи по умолчанию. Возвращаемое значение — дескриптор объекта памяти. Чтобы преобразовать дескриптор в указатель, используйте функцию LockHGlobal . Это значение нельзя объединить с GMEM_FIXED.
GMEM_ZEROINIT	Инициализирует содержимое памяти нулем.

Возвращаемое значение

Указатель или дескриптор выделенной памяти в зависимости от параметра nFlags.

Смотрите также

Ссылки

[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[GlobalAlloc](#)

AllocMem

Выделяет указанное количество байтов из пользовательской библиотечной кучи.

```
AllocMem( nNumberOfBytes )
```

Параметры

nNumberOfBytes

Количество выделяемых байтов.

Возвращаемое значение

Указатель (числовой) на выделенный блок памяти или 0, если произошла ошибка.

Замечания

Выделенная память инициализируется нулем.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapAlloc](#)

AllocMemTo

Выделяет указанное количество байтов из пользовательской библиотечной кучи и сохраняет указатель на выделенную память по переданному адресу.

```
AllocMemTo( nAddress , nNumberOfBytes )
```

Параметры

nAddress

Адрес памяти, где будет сохранен указатель на вновь выделенную память.

nNumberOfBytes

Количество выделяемых байтов.

Возвращаемое значение

Указатель (числовой) на выделенную память или 0, если произошла ошибка.

Замечания

Выделенная память инициализируется нулем.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapAlloc](#)

AMax

Извлекает максимальное значение из массива.

```
AMax( @aArray [, nDimension ])
```

Параметры

@aArray

Массив по ссылке, над которым должно быть выполнено вычисление.

nDimension (необязательно)

по умолчанию = 1

Номер колонки массива, для которой должно быть выполнено вычисление. Если передать 0 в этот параметр, вычисление будет выполнено для всех элементов массива.

Возвращаемое значение

Максимальное из значений, содержащихся в переданном массиве, или .NULL., если все строки были .NULL.

Замечания

Допустимые типы значений в массиве: числовой (N), валюта (Y), дата (D) или дата-время (T).

Все значения в массиве, над которым происходит вычисление, должны быть одного типа, в противном случае возникает ошибка "недопустимые аргументы".

Смотрите также

Ссылки

[AAverage](#)

[AMin](#)

[ASum](#)

AMemBlocks

Сохраняет информацию обо всех блоках, выделенных из внутренней кучи библиотеки, в массиве.

```
AMemBlocks( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [PROCESS_HEAP_ENTRY](#).

Столбец	Значение	Тип данных
1	Указатель на часть данных элемента кучи.	N
2	Размер части данных элемента кучи в байтах.	N
3	Размер данных, используемых системой для хранения информации об элементе кучи, в байтах.	N

Возвращаемое значение

Количество блоков памяти.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapWalk](#)

AMin

Извлекает минимальное значение из массива.

```
AMin( @aArray [, nDimension ])
```

Параметры

@aArray

Массив по ссылке, над которым должно быть выполнено вычисление.

nDimension (необязательно)

по умолчанию = 1

Номер колонки массива, для которой должно быть выполнено вычисление. Если передать 0 в этот параметр, вычисление будет выполнено для всех элементов массива.

Возвращаемое значение

Минимальное из значений, содержащихся в переданном массиве, или .NULL., если все строки были .NULL.

Замечания

Допустимые типы значений в массиве: числовой (N), валюта (Y), дата (D) или дата-время (T).

Все значения в массиве, над которым происходит вычисление, должны быть одного типа, в противном случае возникает ошибка "недопустимые аргументы".

Смотрите также

Ссылки

[AAverage](#)

[AMax](#)

[ASum](#)

ANetFiles

Сохраняет информацию об открытых файлах на сервере в массиве.

```
ANetFiles( cArrayName [, cServer [, cBasepath [, cUser ]]])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [FILE_INFO_3](#).

Столбец	Значение	Тип данных
1	Путь открытого ресурса.	C
2	Строка, указывающая, какой пользователь (на серверах с безопасностью на уровне пользователя) или какой компьютер (на серверах с безопасностью на уровне общего доступа) открыл ресурс.	C
3	Идентификационный номер, присваиваемый ресурсу при его открытии.	N
4	Разрешения доступа, связанные с открываемым приложением.	N
5	Количество блокировок файла, устройства или канала.	N

cServer (необязательно)

по умолчанию = NULL

Имя DNS или NetBIOS удаленного сервера, на котором должна выполняться функция. Если этот параметр равен NULL, используется локальный компьютер.

cBasePath (необязательно)

по умолчанию = NULL

Квалификатор для возвращаемой информации. Если этот параметр равен NULL, перечисляются все открытые ресурсы. Если этот параметр не равен NULL, функция перечисляет только ресурсы, имеющие значение параметра basepath в качестве префикса.

cUser (необязательно)

по умолчанию = NULL

Имя пользователя. Если этот параметр не равен NULL, его значение служит квалификатором для перечисления. Возвращаемые файлы ограничиваются теми, в которых имена пользователей соответствуют квалификатору. Если этот параметр равен NULL, квалификатор имени пользователя не используется.

Возвращаемое значение

Количество файлов.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)
[AIPAddresses](#)
[ANetServers](#)
[GetServerTime](#)
[ICmpPing](#)
[Ip2MacAddress](#)
[ResolveHostToIp](#)

[SyncToSNTPServer](#)
[UrlDownloadToFileEx](#)

Используемые функции WinApi

[NetFileEnum](#)

ANetServers

Сохраняет все серверы указанного типа, видимые в домене, в массиве.

```
ANetServers( cArrayName [, nServerType [, nLevel [, cDomain ]]])
```

Параметры

cArrayName

если *nLevel* = 1

массив содержит следующую информацию из структуры [SERVER_INFO 101](#).

Столбец	Значение	Тип данных
1	Идентификатор платформы	N
2	Имя сервера.	C
3	Основной номер версии и тип сервера.	N
4	Номер младшей версии операционной системы.	N
5	Тип программного обеспечения, установленного на компьютере.	N
6	Комментарий, описывающий сервер.	C

если *nLevel* = 2

массив содержит следующую информацию из структуры [SERVER_INFO 100](#).

Столбец	Значение	Тип данных
1	Идентификатор платформы	C
2	Имя сервера.	C

nServerType (необязательно)

по умолчанию = SV_TYPE_SERVER

Значение, фильтрующее записи сервера. Список возможных значений см. в документации [NetServerEnum](#).

nLevel (необязательно)

по умолчанию = 1

1 или 2.

cDomain (необязательно)

по умолчанию = NULL

Имя домена, для которого должен быть возвращен список серверов. Имя домена должно быть именем домена NetBIOS (например, microsoft). Функция ANetServers не поддерживает имена в стиле DNS (например, microsoft.com).

Возвращаемое значение

Количество серверов.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)
[AIPAddresses](#)
[ANetFiles](#)
[GetServerTime](#)
[ICmpPing](#)
[Ip2MacAddress](#)
[ResolveHostToIp](#)
[SyncToSNTPServer](#)
[UrlDownloadToFileEx](#)

Используемые функции WinApi

[NetServerEnum](#)

APaperSizes

Сохраняет информацию о поддерживаемых размерах бумаги в массиве.

```
APaperSizes( cArrayName , cPrinter , cPort [, nUnit ])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Идентификатор бумаги — см. документацию по члену dmPaperSize структуры DEVMODE	N
2	Имя	C
3	Ширина формата бумаги в требуемых единицах измерения	N
4	Длина формата бумаги в требуемых единицах измерения	N

cPrinter

Название принтера, для которого необходимо перечислить размеры бумаги.

cPort

Имя порта, к которому подключен принтер.

nUnit (необязательно)

по умолчанию = PAPERSIZE_UNIT_MM

Управляет единицей измерения, в которой ширина и длина форматов бумаги сохраняются в массиве.

Возможные значения этого параметра.

Единица	Описание
PAPERSIZE_UNIT_MM	Десятые доли миллиметра
PAPERSIZE_UNIT_INCH	Дюймы
PAPERSIZE_UNIT_POINT	Пункты

Возвращаемое значение

Количество форматов бумаги.

Смотрите также

Ссылки

[APrinterForms](#)

[APrinterForms](#)

[APrinterTrays](#)

[APrintJobs](#)

Используемые функции WinApi

[DeviceCapabilities](#) с параметрами DC_PAPERS, DC_PAPERNAMEs и DC_PAPERSIZE

APrinterForms

Сохраняет информацию о формах, поддерживаемых указанным принтером, в массиве.

```
APrinterForms( cArrayName [, cPrinter ])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [FORM_INFO 1](#).

Столбец	Значение	Тип данных
1	Свойства формы. Список возможных значений смотрите здесь FORM_INFO 1 .	N
2	Название формы.	C
3	Ширина формы в тысячных долях миллиметра.	N
4	Высота формы в тысячных долях миллиметра.	N
5	Нижнее положение печатной области формы, в тысячных долях миллиметра.	N
6	Верхнее положение печатной области формы, в тысячных долях миллиметра.	N
7	Левое положение печатной области формы, в тысячных долях миллиметра.	N
8	Правое положение печатной области формы, в тысячных долях миллиметра.	N

cPrinter

Название принтера, для которого необходимо нумеровать формы.

Возвращаемое значение

Количество печатных форм.

Смотрите также

Ссылки

[APaperSizes](#)
[APrintersEx](#)
[APrinterTrays](#)
[APrintJobs](#)

Используемые функции WinApi

[EnumForms](#)
[OpenPrinter](#)
[ClosePrinter](#)

APrintersEx

Сохраняет информацию о доступных принтерах, серверах печати, доменах или поставщиках печати в массиве.

```
APrintersEx( cArrayName [, cName [, nFlags [, nLevel [,
nOutputType ]]])
```

Параметры

cArrayName

если *nLevel* = 1

массив содержит следующую информацию из структуры [PRINTER_INFO_1](#).

Столбец	Значение	Тип данных
1	Флаги. Посмотрите здесь PRINTER_INFO_1 для списка возможных значений.	N
2	Описывает содержание структуры.	C
3	Называет содержимое структуры.	C
4	Дополнительные данные, описывающие структуру.	C

если *nLevel* = 2

массив содержит следующую информацию из структуры [PRINTER_INFO_2](#).

Столбец	Значение	Тип данных
1	Сервер, который управляет принтером. Если эта строка пуста, принтер управляется локально.	C
2	Название принтера.	C
3	Точка общего доступа для принтера.	C
4	Порт(ы), используемый для передачи данных на принтер.	C
5	Имя драйвера принтера.	C
6	Краткое описание принтера.	C
7	Указывает физическое местоположение принтера.	C
8	Имя файла, используемого для создания страницы-разделителя. Эта страница используется для разделения заданий печати, отправляемых на принтер.	C
9	Название процессора печати, используемого принтером.	C
10	Тип данных, используемый для записи задания печати.	C
11	Параметры процессора печати по умолчанию.	C
12	Атрибуты принтера. Список возможных значений смотрите здесь PRINTER_INFO_2 .	N
13	Значение приоритета, которое спулер использует для маршрутизации заданий печати.	N
14	Значение приоритета по умолчанию, назначаемое каждому заданию печати.	N
15	Самое раннее время, когда принтер напечатает задание. Это значение выражается в минутах, прошедших с 12:00 утра по	N

	Гринвичу (GMT).	
16	Самое позднее время, когда принтер напечатает задание. Это значение выражается в минутах, прошедших с 12:00 по Гринвичу (GMT).	N
17	Состояние принтера. Список возможных значений смотрите здесь PRINTER_INFO_2 .	C
18	Количество заданий на печать, поставленных в очередь для принтера.	N
19	Среднее количество страниц в минуту, напечатанных на принтере.	N

если $nLevel = 4$

массив содержит следующую информацию из структуры [PRINTER_INFO_4](#).

Столбец	Значение	Тип данных
1	Имя принтера (локального или удаленного).	C
2	Имя сервера.	C
3	Атрибуты. Список возможных значений смотрите здесь PRINTER_INFO_4 .	N

если $nLevel = 5$

массив содержит следующую информацию из структуры [PRINTER_INFO_5](#).

Столбец	Значение	Тип данных
1	Название принтера.	C
2	Порт(ы), используемый для передачи данных на принтер.	C
3	Атрибуты принтера. Список возможных значений смотрите здесь PRINTER_INFO_5 .	N
4	DeviceNotSelectedTimeout	N
5	TransmissionRetryTimeout	N

cName (необязательно)

по умолчанию = empty

Имя процессора печати, домена или сервера, для которого необходимо перечислить принтеры.

Если $nLevel$ равен 1, $nFlags$ содержит PRINTER_ENUM_NAME, а $cName$ не пуст, то $cName$ указывает имя объекта для перечисления.

Эта строка может быть именем сервера, домена или поставщика печати.

Если $nLevel$ равен 1, $nFlags$ содержит PRINTER_ENUM_NAME, а $cName$ пуст, то функция перечисляет доступных поставщиков печати.

Если $nLevel$ равен 1, $nFlags$ содержит PRINTER_ENUM_REMOTE, а $cName$ пуст, то функция перечисляет принтеры в домене пользователя.

Если $nLevel$ равен 2 или 5, $cName$ указывает имя сервера, принтеры которого необходимо перечислить. Если $cName$ пуст, то функция перечисляет принтеры, установленные на локальном компьютере.

Если *nLevel* равен 4, *cName* должен быть пустым. Функция всегда запрашивает данные на локальной машине.

Если *cName* пустое, установка *nFlags* в PRINTER_ENUM_LOCAL + PRINTER_ENUM_CONNECTIONS перечисляет принтеры, установленные на локальной машине. Эти принтеры включают те, которые физически подключены к локальной машине, а также удаленные принтеры, к которым она имеет сетевое подключение.

***nFlags* (необязательный, дополнительный)**

по умолчанию = PRINTER_ENUM_LOCAL

Одно или комбинация следующих значений.

Флаг	Описание
PRINTER_ENUM_LOCAL	Функция игнорирует параметр <i>cName</i> и перечисляет локально установленные принтеры. Windows 95/98/Me: Функция также перечислит сетевые принтеры, поскольку они обрабатываются локальным поставщиком печати.
PRINTER_ENUM_NAME	Функция перечисляет принтер, идентифицированный <i>cName</i> . Это может быть сервер, домен или поставщик печати. Если <i>cName</i> пусто, функция перечисляет доступные поставщики печати.
PRINTER_ENUM_SHARED	Функция перечисляет принтеры, имеющие атрибут <i>shared</i> . Не может использоваться изолированно; используйте операцию + для объединения с другим типом PRINTER_ENUM.
PRINTER_ENUM_DEFAULT	Windows 95/98/Me: Функция возвращает информацию о принтере по умолчанию.
PRINTER_ENUM_CONNECTIONS	Windows NT/2000/XP: Функция перечисляет список принтеров, к которым пользователь ранее подключался.
PRINTER_ENUM_NETWORK	Windows NT/2000/XP: Функция перечисляет сетевые принтеры в домене компьютера. Это значение действительно, только если <i>nLevel</i> равен 1.
PRINTER_ENUM_REMOTE	Windows NT/2000/XP: Функция перечисляет сетевые принтеры и серверы печати в домене компьютера. Это значение действительно, только если <i>nLevel</i> равен 1.

***nLevel* (необязательно)**

по умолчанию = 2

Указывает уровень информации, возвращаемой функцией.

Допустимые значения: 1, 2, 4 и 5, которые соответствуют структурам данных [PRINTER_INFO_1](#), [PRINTER_INFO_2](#), [PRINTER_INFO_4](#) и [PRINTER_INFO_5](#).

Windows 95/98/Me: Значение может быть 1, 2 или 5. Windows NT/2000/XP: Это значение может быть 1, 2, 4 или 5.

***nOutputType* (необязательно)**

по умолчанию = APRINT_OUTPUT_ARRAY

Допустимые значения.

APRINT_OUTPUT_ARRAY — информация возвращается в многомерном массиве.

APRINT_OUTPUT_OBJECTARRAY — информация возвращается в одномерном массиве, где каждая строка содержит объект с пользовательским набором свойств.

Возвращаемое значение

Количество принтеров.

Смотрите также

Ссылки

[APaperSizes](#)

[APrinterForms](#)

[APrinterTrays](#)

[APrintJobs](#)

Используемые функции WinApi

[EnumPrinters](#)

APrinterTrays

Сохраняет информацию о доступных лотках для бумаги принтера в массиве.

```
APrinterTrays( cArrayName , cPrinter , cPort )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Название лотка принтера.	C
2	Идентификатор лотка.	N

cPrinter

Название принтера, для которого необходимо пронумеровать лотки.

cPort

Имя порта, к которому подключен принтер.

Возвращаемое значение

Количество лотков принтера.

Смотрите также

Ссылки

[APaperSizes](#)

[APrinterForms](#)

[APrintersEx](#)

[APrintJobs](#)

Используемые функции WinApi

[DeviceCapabilities](#) с параметрами DC_BINS и DC_BINNAMES

APrintJobs

Сохраняет информацию об указанном наборе заданий печати для указанного принтера в массиве.

```
APrintJobs( cArrayName , cPrinter [, nLevel ] )
```

Параметры

cArrayName

если *nLevel* = 1

массив содержит следующую информацию из структуры [JOB_INFO_1](#).

Столбец	Значение	Тип данных
1	Название задания на печать.	C
2	Имя принтера, для которого задано задание.	C
3	Имя пользователя, которому принадлежит задание печати.	C
4	Имя машины, создавшей задание печати.	C
5	Тип данных, используемых для записи задания печати.	C
6	Статус задания печати.	C
7	Идентификатор работы.	N
8	Статус задания. Список возможных значений смотрите здесь JOB_INFO_1 .	N
9	Приоритет задания. Список возможных значений смотрите здесь JOB_INFO_1 .	N
10	Положение задания в очереди печати.	N
11	Общее количество страниц, содержащихся в документе. Это значение может быть равно нулю, если задание на печать не содержит информации о разграничении страниц.	N
12	Количество напечатанных страниц. Это значение может быть равно нулю, если задание на печать не содержит информации о разграничении страниц.	N
13	Время, когда этот документ был загружен. Это значение времени указано в формате всемирного времени (UTC). Перед отображением его следует преобразовать в местное время. Для выполнения преобразования можно использовать функцию UTC2DT .	T

если *nLevel* = 2

массив содержит следующую дополнительную информацию из структуры [JOB_INFO_2](#).

Столбец	Значение	Тип данных
14	Имя пользователя, которого следует уведомить о завершении печати задания или о возникновении ошибки во время печати задания.	C
15	Название процессора печати, который следует использовать для печати задания.	C
16	Параметры процессора печати.	C

17	Имя драйвера принтера, который следует использовать для обработки задания печати.	C
18	Общее время в миллисекундах, прошедшее с момента начала печати задания.	N
19	Самый ранний срок, когда задание может быть напечатано.	N
20	Самый поздний срок, когда задание может быть напечатано.	N
21	Размер задания в байтах.	N

cPrinter

Имя принтера, для которого необходимо перечислить задания печати.

nLevel (необязательно)

по умолчанию = 1

Уровень возвращаемой информации: 1 или 2.

Возвращаемое значение

Количество заданий на печать.

Смотрите также**Ссылки**

[APaperSizes](#)
[APrinterForms](#)
[APrintersEx](#)
[APrinterTrays](#)

Используемые функции WinApi

[EnumJobs](#)
[OpenPrinter](#)
[ClosePrinter](#)

AProcesses

Сохраняет информацию о процессах, запущенных в данный момент в системе, в массиве.

```
AProcesses( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [PROCESSENTRY32](#).

Столбец	Значение	Тип данных
1	Имя исполняемого файла	C
2	Идентификатор процесса	N
3	Идентификатор родительского процесса	N
4	Количество потоков	N
5	Базовый класс приоритета	N

Возвращаемое значение

Количество процессов.

Смотрите также

Ссылки

[AHeapBlocks](#)
[AProcessHeaps](#)
[AProcessModules](#)
[AProcessThreads](#)

Используемые функции WinApi

[CreateToolhelp32Snapshot](#)
[Process32First](#)
[Process32Next](#)
[EnumProcesses](#) (только для Windows NT)
[OpenProcess](#) (только для Windows NT)
[EnumProcessModules](#) (только для Windows NT)
[NtQueryInformationProcess](#) (только для Windows NT)
[CloseHandle](#) (только для Windows NT)

AProcessHeaps

Сохраняет информацию о кучах процесса в массиве.

```
AProcessHeaps( cArrayName , nProcessID )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [HEAPLIST32](#).

Столбец	Значение	Тип данных
1	Идентификатор кучи	N
2	Флаги	N

nProcessID

Идентификатор процесса, для которого необходимо перечислить кучи.

Возвращаемое значение

Количество куч процессов.

Смотрите также

Ссылки

[AHeapBlocks](#)

[AProcess](#)

[AProcessModules](#)

[AProcessThreads](#)

Используемые функции WinApi

[CreateToolhelp32Snapshot](#)

[Heap32ListFirst](#)

[Heap32ListNext](#)

AProcessModules

Сохраняет информацию о модулях (DLL), загруженных процессом, в массиве.

```
AProcessModules( cArrayName , nProcessID )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [MODULEENTRY32](#).

Столбец	Значение	Тип данных
1	Модуль	C
2	Путь к исполняемому файлу	C
3	Дескриптор модуля	N
4	Базовый размер	N
5	Базовый адрес	N

nProcessID

Идентификатор процесса, для которого необходимо перечислить загруженные модули.

Возвращаемое значение

Количество модулей.

Смотрите также

Ссылки

[AHeapBlocks](#)
[AProcesses](#)
[AProcessHeaps](#)
[AProcessThreads](#)

Используемые функции WinApi

[CreateToolhelp32Snapshot](#)
[Module32First](#)
[Module32Next](#)

AProcessThreads

Сохраняет информацию о потоках процесса в массиве.

```
AProcessThreads( cArrayName , nProcessID )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [THREADENTRY32](#).

Столбец	Значение	Тип данных
1	Идентификатор потока	N
2	Идентификатор процесса владельца	N
3	Базовый приоритет	N

nProcessID

Идентификатор процесса, для которого необходимо перечислить потоки.

Возвращаемое значение

Количество потоков.

Смотрите также

Ссылки

[AHeapBlocks](#)
[AProcess](#)
[AProcessHeaps](#)
[AProcessModules](#)

Используемые функции WinApi

[CreateToolhelp32Snapshot](#)
[Thread32First](#)
[Thread32Next](#)

ARasConnections

Сохраняет информацию обо всех активных RAS-подключениях в массиве.

```
ARasConnections( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структур [RASCONN](#) и [RASPPPIP](#).

Столбец	Значение
1	Имя коммутируемого соединения.
2	Имя устройства.
3	Тип устройства.
4	IP-адрес.
5	Дескриптор HRASCONN.

Возвращаемое значение

Количество RAS-подключений.

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)

[RasConnectionNotificationEx](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasGetConnectStatusEx](#)

[RasHangUpEx](#)

[RasPhonebookDlgEx](#)

Используемые функции WinApi

[RasEnumConnections](#)

[RasGetProjectionInfo](#)

ARasDevices

Сохраняет информацию обо всех доступных устройствах с поддержкой RAS в массиве.

```
ARasDevices( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [RASDEVINFO](#).

Столбец	Значение
1	Тип устройства.
2	Название устройства.

Возвращаемое значение

Количество устройств RAS.

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)
[ARasConnections](#)
[ARasPhonebookEntries](#)
[RasClearConnectionStatistics](#)
[RasConnectionNotificationEx](#)
[RasDialDlgEx](#)
[RasDialEx](#)
[RasGetConnectStatusEx](#)
[RasHangUpEx](#)
[RasPhonebookDlgEx](#)

Используемые функции WinApi

[RasEnumDevices](#)

ARasPhonebookEntries

Сохраняет информацию обо всех именах записей в телефонной книге удаленного доступа в массиве.

```
ARasPhonebookEntries( cArrayName [, cPhonebookfile ] )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [RASENTRYNAME](#).

Столбец	Значение
1	Имя записи телефонной книги.
2	Флаги записи телефонной книги.

Значение флагов может быть: REN_User или REN_AllUsers (определено в rasapi32.h).

В Windows 9x флаги не поддерживаются и поэтому всегда равны 0.

cPhonebookfile (необязательно)

по умолчанию = используется системная телефонная книга по умолчанию,

в противном случае передайте допустимое имя файла телефонной книги.

Возвращаемое значение

Количество записей в телефонной книге RAS.

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)

[ARasConnections](#)

[ARasDevices](#)

[RasClearConnectionStatistics](#)

[RasConnectionNotificationEx](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasGetConnectStatusEx](#)

[RasHangUpEx](#)

[RasPhonebookDlgEx](#)

Используемые функции WinApi

[RasEnumEntries](#)

ARegistryKeys

Сохраняет информацию обо всех подразделах указанного раздела реестра в массиве.

```
ARegistryKeys( cArrayName , nRegKey , cKeyName [, nFlags ] )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Имя ключа реестра	C
2	Имя класса ключа реестра, только если <i>nFlags</i> содержит REG_ENUMCLASSNAME	C
3	Время последней записи, только если <i>nFlags</i> содержит REG_ENUMWRITETIME	T

nRegKey

Либо дескриптор реестра, возвращенный [OpenRegistryKey](#), либо одна из следующих констант ключа.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего</p>

	<p>пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и устанавливается при первой ссылке процесса на HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их</p>

	источника.
HKEY_PERFORMANCE_NLSTEXT	Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe. Windows 2000: этот ключ не поддерживается.
HKEY_PERFORMANCE_TEXT	Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe. Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

sKeyName (необязательно)

Путь к подключаемому ключу ключа, переданному в *nRegKey*.

nFlags (необязательно)

по умолчанию = 0

Комбинация следующих значений.

Флаг	Значение
REG_ENUMCLASSNAME	Также извлекается имя класса ключа реестра.
REG_ENUMWRITETIME	Также извлекается время последней записи в реестр.

Возвращаемое значение

Количество ключей реестра.

Смотрите также**Ссылки**

[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegEnumKeyEx](#)
[RegOpenKeyEx](#)
[RegQueryInfoKey](#)

ARegistryValues

Сохраняет информацию обо всех значениях указанного раздела реестра в массиве.

```
ARegistryValues( cArrayName , nRegKey , cKeyName [, nFlags ])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Имя значения реестра	C
2	Тип значения реестра, только если nFlags содержит REG_ENUMTYPE	C
3	Данные для значения реестра, только если nFlags содержит REG_ENUMVALUE	variant

nRegKey

Либо дескриптор реестра, возвращенный [OpenRegistryKey](#), либо одна из следующих констант ключа.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER</p>

	<p>поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и устанавливается при первой ссылке процесса на HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>

HKEY_PERFORMANCE_NLSTEXT	Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe. Windows 2000: этот ключ не поддерживается.
HKEY_PERFORMANCE_TEXT	Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe. Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

cKeyName (необязательно)

Путь к подключаемому ключу ключа, переданному в *nRegKey*.

nFlags (необязательно)

по умолчанию = 0

Комбинация следующих значений.

Флаг	Значение
REG_ENUMTYPE	Также извлекается тип значения реестра.
REG_ENUMVALUE	Также извлекаются данные значения реестра.

Возвращаемое значение

Количество значений реестра.

Смотрите также**Ссылки**

[ARegistryKeys](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegEnumValue](#)
[RegOpenKeyEx](#)
[RegQueryInfoKey](#)

AResolutions

Сохраняет информацию обо всех графических режимах (разрешениях) для устройства отображения в массиве.

```
AResolutions( cArrayName [, cDeviceName ] )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение
1	Указывает ширину разрешения в пикселях.
2	Указывает высоту разрешения в пикселях.
3	Указывает цветовое разрешение в битах на пиксель.
4	Указывает частоту в герцах (циклах в секунду).

cDeviceName (необязательно)

по умолчанию = empty

Этот параметр либо пуст, либо имя устройства отображения, возвращаемое из [ADisplayDevices](#).

Пустое значение указывает текущее устройство отображения на компьютере, на котором запущен вызывающий поток.

Возвращаемое значение

Количество разрешений.

Смотрите также

Ссылки

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AWindowStations](#)

[ExpandEnvironmentStringsEx](#)

[GetLocaleInfoEx](#)

[GetSystemDirectoryEx](#)

[GetWindowsDirectoryEx](#)

[OsEx](#)

Используемые функции WinApi

[EnumDisplaySettings](#)

AResourceLanguages

Сохраняет информацию о языковых ресурсах указанного типа и имени, связанных с двоичным модулем, в массиве.

```
AResourceLanguages( cArrayName , nModule , cnResourceType ,  
cnResourceName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Идентификатор языка ресурса.	N

nModule

Дескриптор модуля для поиска. Начиная с Windows Vista, если это независимый от языка Portable Executable (файл LN), то в поиск включаются соответствующие файлы .mui (если таковые имеются). Если это определенный файл .mui, поиск ресурсов выполняется только в этом файле.

Если этот параметр равен 0, это эквивалентно передаче дескриптора модулю, используемому для создания текущего процесса.

cnResourceType

Тип ресурса, для которого перечисляется язык.

В качестве альтернативы, вместо строки, этот параметр может быть целочисленным значением, представляющим predetermined тип ресурса.

cnResourceName

Имя ресурса, для которого перечисляется язык.

В качестве альтернативы, вместо строки, этот параметр может быть целочисленным идентификатором ресурса.

Возвращаемое значение

Количество языков ресурсов.

Смотрите также

Ссылки

[AResourceNames](#)

[AResourceTypes](#)

Используемые функции WinApi

[EnumResourceLanguages](#)

AResourceNames

Сохраняет информацию о ресурсах указанного типа в двоичном модуле в массиве.

```
AResourceNames( cArrayName , nModule , cnResourceType )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Имя или идентификатор ресурса.	С или N

nModule

Дескриптор модуля для поиска. Начиная с Windows Vista, если это независимый от языка Portable Executable (файл LN), то в поиск включаются соответствующие файлы .mui (если таковые имеются). Если это определенный файл .mui, поиск ресурсов выполняется только в этом файле.

Если этот параметр равен 0, это эквивалентно передаче дескриптора модулю, используемому для создания текущего процесса.

cnResourceType

Тип ресурса, для которого перечисляется язык.

В качестве альтернативы, вместо строки, этот параметр может быть целочисленным значением, представляющим predetermined тип ресурса.

Возвращаемое значение

Количество названий ресурсов.

Смотрите также

Ссылки

[AResourceLanguages](#)

[AResourceTypes](#)

Используемые функции WinApi

[EnumResourceNames](#)

AResourceTypes

Сохраняет информацию о типах ресурсов внутри двоичного модуля в массиве.

```
AResourceTypes( cArrayName , nHMODULE )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Имя или идентификатор типа ресурса.	С или N

nModule

Дескриптор модуля для поиска. Начиная с Windows Vista, если это независимый от языка Portable Executable (файл LN), то в поиск включаются соответствующие файлы .mui (если таковые имеются). Если это определенный файл .mui, поиск ресурсов выполняется только в этом файле.

Если этот параметр равен 0, это эквивалентно передаче дескриптора модулю, используемому для создания текущего процесса.

Возвращаемое значение

Количество типов ресурсов.

Смотрите также

Ссылки

[AResourceLanguages](#)

[AResourceNames](#)

Используемые функции WinApi

[EnumResourceTypes](#)

AServiceConfig

Сохраняет параметры конфигурации указанной службы Windows в массиве.

```
AServiceConfig( cArrayName , cServiceName | nServiceHandle [,  
cServer [, cDatabase ]])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [QUERY_SERVICE_CONFIG](#).

Элемент	Значение
1	ServiceType
2	StartType
3	ErrorControl
4	BinaryPathName
5	LoadOrderGroup
6	TagId
7	Dependencies
8	ServiceStartName
9	DisplayName

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

cServer (необязательно)

Имя сервера, на котором запущена служба.
См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.
См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADependentServices](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[QueryServiceConfig](#)
[OpenSCManager](#)
[OpenService](#)
[CloseServiceHandle](#)

AServices

Сохраняет информацию о службах Windows в массиве.

```
AServices( cArrayName , [ cServer [, cDatabase [, nServiceState [, nServiceType ]]] ] )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [ENUM_SERVICE_STATUS_PROCESS](#) или [ENUM_SERVICE_STATUS](#).

Столбец	Значение
1	ServiceName
2	DisplayName
3	ServiceType
4	CurrentState
5	Win32ExitCode
6	ServiceSpecificExitCode
7	CheckPoint
8	ControlsAccepted
9	ServiceFlags
10	ProcessId

cServer (необязательно)

Имя сервера, на котором запущена служба.
См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.
См. справку MSDN для [OpenSCManager](#).

nServiceState (необязательно)

по умолчанию = SERVICE_STATE_ALL

Одно из следующих значений.

ServiceState	Описание
SERVICE_ACTIVE	Перечислить только запущенные службы.
SERVICE_INACTIVE	Перечислить только остановленные службы.
SERVICE_STATE_ALL	Перечислите все услуги.

nServiceType (необязательно, добавочный)

по умолчанию = SERVICE_WIN32

Одно или комбинация следующих значений.

ServiceType	Описание
SERVICE_DRIVER	Перечисляет службы типа SERVICE_KERNEL_DRIVER и SERVICE_FILE_SYSTEM_DRIVER.
SERVICE_WIN32	Перечисляет службы типа SERVICE_WIN32_OWN_PROCESS и

SERVICE_WIN32_SHARE_PROCESS.

Возвращаемое значение

Количество служб.

Смотрите также

Ссылки

[ADependentServices](#)
[AServiceConfig](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[EnumServicesStatusEx](#), если поддерживается, в противном случае
[EnumServicesStatus](#)
[OpenSCManager](#)
[CloseServiceHandle](#)

AServiceStatus

Сохраняет информацию о текущем состоянии указанной службы в массиве.

```
AServiceStatus( cArrayName , cServiceName | nServiceHandle [,  
cServer [, cDatabase ]])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию из структуры [SERVICE_STATUS](#).

Элемент	Значение
1	CheckPoint
2	ControlsAccepted
3	Currentstate
4	ServiceSpecificExitCode
5	Servicetype
6	WaitHint
7	Win32ExitCode

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

cServer (необязательно)

Имя сервера, на котором запущена служба.
См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.
См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[QueryServiceStatus](#)
[OpenSCManager](#)

[OpenService](#)[CloseServiceHandle](#)

ASplitStr

Разбивает строку на массив на основе указанной длины.

```
ASplitStr( cArrayName , cString , nLen )
```

Параметры

cArrayName

Имя массива, на который следует разбить строку.

cСтрока

Строка, которую нужно разделить.

nLen

Длина каждой подстроки.

Возвращаемое значение

Количество подстрок.

Пример

```
LOCAL lcString, lnCount, laSubString[1]
m.lcString = '12345678901234567890123456789012345'
m.lnCount = ASplitStr('laSubString', m.lcString, 10)
DISPLAY MEMORY LIKE laSubString
```

Смотрите также

Ссылки

[AFontInfo](#)
[Decimals](#)
[GetCursorPosEx](#)
[Int64_Add](#)
[Int64_Div](#)
[Int64_Mod](#)
[Int64_Mul](#)
[Int64_Sub](#)

ASQLDataSources

Сохраняет информацию о настроенных источниках данных ODBC в массиве.

```
ASQLDataSources( cArrayName [, nDataSourceType ])
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Имя источника данных.	C
2	Описание источника данных.	C

nDataSourceType (необязательно)

по умолчанию = ODBC_BOTH_DSN

Одно из следующих значений.

Тип источника данных	Описание
ODBC_BOTH_DSN	Перечисляет как пользовательские, так и системные источники данных.
ODBC_USER_DSN	Перечисляет только пользовательские источники данных.
ODBC_SYSTEM_DSN	Перечисляет только системные источники данных.

Возвращаемое значение

Количество источников данных SQL.

Смотрите также

Ссылки

[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLDataSources](#)

ASQLDrivers

Сохраняет информацию об установленных драйверах ODBC в массиве.

```
ASQLDrivers( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Описание драйвера.	C
2	Список пар значений атрибутов драйвера, каждая запись разделена символом CHR (0).	C

Возвращаемое значение

Количество драйверов SQL.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLDrivers](#)

ASum

Вычисляет сумму значений в массиве.

```
ASum( @aArray [, nDimension ])
```

Параметры

@aArray

Массив по ссылке, над которым должно быть выполнено вычисление.

nDimension (необязательно)

по умолчанию = 1

Номер колонки массива, для которой должно быть выполнено вычисление. Если передать 0 в этот параметр, вычисление будет выполнено для всех элементов массива.

Возвращаемое значение

Сумма значений, содержащихся в переданном массиве, или .NULL., если все строки были .NULL.

Замечания

Допустимые типы значений в массиве: числовой (N) или денежный (Y). Все значения в массиве, над которым выполняется вычисление, должны быть одного типа, в противном случае возникает ошибка "недопустимые аргументы".

Если тип значений — Y, а сумма всех значений не попадает в диапазон денежного типа данных, возникает ошибка 1988 "Значение валюты выходит за пределы диапазона".

Смотрите также

Ссылки

[AAverage](#)

[AMax](#)

[AMin](#)

AsyncWaitForObject

Ожидает API HANDLE в отдельном потоке без блокировки; при получении сигнала HANDLE выполняется предоставленная функция обратного вызова.

```
AsyncWaitForObject( nHandle , cCallback )
```

Параметры

nHandle

Дескриптор для мониторинга.

cCallback

Функция, которая будет вызвана при сигнале дескриптора.
Функция должна иметь следующий прототип:

```
FUNCTION ObjectSignaled
    LPARAMETERS nError
    IF nError = 0
        ? 'Объект получил сигнал'
    ELSE
        ? 'Функция: WaitForMultipleObjects завершилась '+ ;
        'ошибкой', nError
    ENDIF
ENDFUNC
```

Возвращаемое значение

Дескриптор (числовой) созданного потока.

Смотрите также

Ссылки

[BindEventsEx](#)
[CancelWaitForObject](#)
[CreateCallbackFunc](#)
[CreatePublicShadowObjReference](#)
[DestroyCallbackFunc](#)
[ReleasePublicShadowObjReference](#)
[UnbindEventsEx](#)

Используемые функции WinApi

[WaitForMultipleObjects](#)
[CreateEvent](#)
[CreateThread](#)
[CloseHandle](#)
[PostMessage](#)

ATimeZones

Извлекает информацию о часовых поясах.

```
ATimeZones( cArrayName )
```

Параметры

cArrayName

При возврате массив содержит информацию из следующего раздела реестра "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Time Zones". См. также описание структуры [TIME_ZONE_INFORMATION](#).

Столбец	Значение
1	Обобщённое название
2	Отображаемое имя часового пояса
3	Стандартное название часового пояса
4	Название часового пояса летнего времени
5	Bias: смещение для перевода времени в минутах. Смещение — это разница в минутах между всемирным координированным временем (UTC) и местным временем. Все переводы между UTC и местным временем основаны на следующей формуле: UTC = местное время + смещение
6	StandardBias: Значение смещения, которое будет использоваться при переводе местного времени, которое происходит во время стандартного времени. Этот член игнорируется, если значение для члена StandardDate не указано. Это значение добавляется к значению члена Bias для формирования смещения, используемого во время стандартного времени. В большинстве часовых поясов значение этого члена равно нулю.
7	DaylightBias: Значение смещения, которое будет использоваться при переводе местного времени во время летнего времени. Этот член игнорируется, если значение для члена DaylightDate не указано. Это значение добавляется к значению члена Bias для формирования смещения, используемого во время летнего времени. В большинстве часовых поясов значение этого члена равно -60.
	StandardDate: Дата и местное время перехода с летнего времени на зимнее в данной операционной системе.
8	Месяц StandardDate
9	День StandardDate
10	День недели StandardDate
11	Час StandardDate
	DaylightDate: Дата и местное время перехода со стандартного времени на летнее время в данной операционной системе.
12	Месяц DaylightDate
13	День DaylightDate
14	День недели DaylightDate
15	Час DaylightDate

Возвращаемое значение

Количество часовых поясов.

Смотрите также

Ссылки

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

Используемые функции WinApi

[RegOpenKeyEx](#)

[RegQueryInfoKey](#)

[RegEnumKeyEx](#)

[RegQueryValueEx](#)

[RegCloseKey](#)

AVolumeInformation

Сохраняет информацию о файловой системе и томе, связанных с указанным корневым каталогом, в массиве.

```
AVolumeInformation( cArrayName , cRootPath )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Элемент	Значение	Тип данных
1	Название указанного тома.	N
2	Серийный номер тома.	N
3	Максимальная длина компонента имени файла в символах, поддерживаемая указанной файловой системой.	N
4	Флаги, связанные с указанной файловой системой. Список возможных значений см. в GetVolumeInformation .	C
5	Файловая система, например, файловая система FAT или файловая система NTFS.	C

cRootPath

Корневой каталог описываемого тома.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AVolumeMountPoints](#)

[AVolumePaths](#)

[AVolumes](#)

Используемые функции WinApi

[GetVolumeInformation](#)

[SetErrorMode](#)

AVolumeMountPoints

Сохраняет имена смонтированных папок на указанном томе в массиве.

```
AVolumeMountPoints( cArrayName , cVolume )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Имя смонтированной папки.	C

cVolume

Путь GUID тома для сканирования тома на предмет смонтированных папок.
Список путей GUID тома можно получить с помощью [AVolumes](#).

Возвращаемое значение

Количество точек монтирования тома.

Смотрите также

Ссылки

[AVolumeInformation](#)
[AVolumePaths](#)
[AVolumes](#)

Используемые функции WinApi

[FindFirstVolumeMountPoint](#)
[FindNextVolumeMountPoint](#)
[FindVolumeMountPointClose](#)

AVolumePaths

Сохраняет буквы дисков и пути GUID томов для указанного тома в массиве.

```
AVolumePaths( cArrayName , cVolume )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Буква диска или пути GUID тома.	C

cVolume

Путь к GUID тома.

Список путей GUID тома можно получить с помощью [AVolumes](#).

Возвращаемое значение

Количество путей томов.

Смотрите также

Ссылки

[AVolumeInformation](#)
[AVolumeMountPoints](#)
[AVolumes](#)

Используемые функции WinApi

[GetVolumePathNamesForVolumeName](#)

AVolumes

Сохраняет имена томов на компьютере в массиве.

```
AVolumes( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Путь GUID тома.	C

Возвращаемое значение

Количество томов.

Смотрите также

Ссылки

[AVolumeInformation](#)
[AVolumeMountPoints](#)
[AVolumePaths](#)

Используемые функции WinApi

[FindFirstVolume](#)
[FindNextVolume](#)
[FindVolumeClose](#)

AWindowProps

Сохраняет информацию о записях в списке свойств окна в массиве.

```
AWindowProps( cArrayName , hWndd )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Имя свойства.	C
2	Дескриптор данных.	N

nHwnd

Дескриптор окна, для которого необходимо получить свойства.

Возвращаемое значение

Количество свойств окна.

Смотрите также

Ссылки

[AWindows](#)

[AWindowsEx](#)

[CenterWindowEx](#)

[GetWindowRectEx](#)

[GetWindowTextEx](#)

Используемые функции WinApi

[EnumPropsEx](#)

AWindows

Сохраняет дескрипторы окон (HWND) в массиве.

```
AWindows( cArrayName , nType [, nParam ] )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Дескриптор окна (HWND).	N

nType (добавочный)

Одно из следующих значений.

Тип	Описание
AWINDOWS_TOPLEVEL	Возвращает окна верхнего уровня.
AWINDOWS_CHILD	Возвращает дочерние окна окна, переданного в nParam.
AWINDOWS_THREAD	Возвращает окна, принадлежащие потоку, переданному в nParam.
AWINDOWS_DESKTOP	Возвращает окна, содержащиеся на рабочем столе, переданном в nParam.
AWINDOWS_CALLBACK	Интерпретировать первый параметр как функцию обратного вызова (добавочную к вышеприведенной).

nParam

В зависимости от параметра nType этот параметр должен иметь одно из следующих значений.

Тип	Значение этого параметра
AWINDOWS_TOPLEVEL	Не передавайте этот параметр.
AWINDOWS_CHILD	Дескриптор окна, для которого необходимо получить дочерние окна.
AWINDOWS_THREAD	Дескриптор потока, для которого необходимо получить принадлежащие ему окна.
AWINDOWS_DESKTOP	Дескриптор рабочего стола, для которого необходимо извлечь содержащиеся окна.

Возвращаемое значение

Если nType содержит AWINDOWS_CALLBACK, возвращается 1, в противном случае — количество окон.

Пример

Перечислить окна верхнего уровня в массив:

```
AWindows('laArray',1)
```

Перечислите окна верхнего уровня, вызвав SomeFunction с параметром hWnd для каждого найденного окна:

```
AWindows('SomeFunction',1+16)
```

Перечислить дочерние окна _SCREEN:

```
AWindows('laArray',2,_SCREEN.hWnd)
```

Перечислить окна потока:

```
AWindows('laArray',4,_VFP.ThreadId)
```

Смотрите также

Ссылки

[AWindowProps](#)

[AWindowEx](#)

[CenterWindowEx](#)

[GetWindowRectEx](#)

[GetWindowTextEx](#)

Используемые функции WinApi

[EnumWindows](#),

если nType = AWINDOWS_TOPLEVEL

[EnumChildWindows](#),

если nType = AWINDOWS_CHILD

[EnumThreadWindows](#),

если nType = AWINDOWS_THREAD

[EnumDesktopWindows](#),

если nType = AWINDOWS_DESKTOP

AWindowsEx

Сохраняет информацию об окнах в массиве.

```
AWindowsEx( cArrayName , cFlags , nType [, nParam ] )
```

Параметры

cFlags (добавочный)

Позиция символа определяет позицию столбца в результирующем массиве.

Символ	Информация
W	HWND
C	Class
T	Text
S	Style
E	ExStyle
H	HInstance
P	HWND родительского окна
D	Пользовательские данные
I	ID
R	ThreadID
O	ProcessID
V	Visible
N	Iconic
M	Maximized
U	Unicode

nType

Одно из следующих значений.

Тип	Описание
AWINDOWS_TOPLEVEL	Возвращает окна верхнего уровня.
AWINDOWS_CHILD	Возвращает дочерние окна окна, переданного в nParam.
AWINDOWS_THREAD	Возвращает окна, принадлежащие потоку, переданному в nParam.
AWINDOWS_DESKTOP	Возвращает окна, содержащиеся на рабочем столе, переданном в nParam.

nParam (необязательно)

В зависимости от параметра nType этот параметр должен иметь одно из следующих значений.

Тип	Значение этого параметра
AWINDOWS_TOPLEVEL	Не передавайте этот параметр.
AWINDOWS_CHILD	Дескриптор окна, для которого необходимо получить дочерние окна.
AWINDOWS_THREAD	Дескриптор потока, для которого необходимо получить принадлежащие ему окна.

AWINDOWS_DESKTOP

Дескриптор рабочего стола, для которого необходимо извлечь содержащиеся окна.

Возвращаемое значение

Количество окон.

Смотрите также

Ссылки

[AWindowProps](#)
[AWindows](#)
[CenterWindowEx](#)
[GetWindowRectEx](#)
[GetWindowTextEx](#)

Используемые функции WinApi

EnumWindows ,	если nType = AWINDOWS_TOPLEVEL
EnumChildWindows ,	если nType = AWINDOWS_CHILD
EnumThreadWindows ,	если nType = AWINDOWS_THREAD
EnumDesktopWindows ,	если nType = AWINDOWS_DESKTOP
GetClassName	
GetWindowText	
GetWindowLong	
GetParent	
GetWindowThreadProcessId	
IsWindowVisible	
IsIconic	
IsZoomed	
IsWindowUnicode	

AWindowStations

Сохраняет имена всех оконных станций в текущем сеансе в массив.

```
AWindowStations( cArrayName )
```

Параметры

cArrayName

По возвращении массив содержит следующую информацию.

Столбец	Значение	Тип данных
1	Название оконной станции.	C

Возвращаемое значение

Смотрите также

Ссылки

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[ExpandEnvironmentStringsEx](#)

[GetLocaleInfoEx](#)

[GetSystemDirectoryEx](#)

[GetWindowsDirectoryEx](#)

[OsEx](#)

Используемые функции WinApi

[EnumWindowStations](#)

BindEventsEx

Предоставляет возможность выполнить функцию или метод объекта, когда окно API получает указанное сообщение окна.

```
BindEventsEx( hWnd , nMsg , oObject , cFunction [,  
cParmDefinition [, nFlags ]])
```

Параметры

nHwnd

Дескриптор окна, сообщения которого следует перехватывать.

nMsg

Сообщение окна, которое должно вызвать функцию обратного вызова, например WM_MOUSEMOVE, WM_ACTIVATE

oObject

должна быть вызвана функция, переданная в *cFunction*, или *NULL*, если *cFunction* является публичной ФУНКЦИЕЙ/ПРОЦЕДУРОЙ.

cFunction

Имя функции или метода, который следует вызвать при получении сообщения окном.

cParmDefinition (необязательно)

Разделенный запятыми список параметров для передачи в функцию.

Если вы опустите cParmDefinition или передадите NULL, то значение по умолчанию будет как в BINDEVENTSEX, то есть в вашу функцию обратного вызова передаются 4 параметра:

```
FUNCTION YourFunc(hWnd, uMsg, wParam, lParam)
```

и wParam & lParam маршалируются как целые числа со знаком.

Вы также можете указать свое собственное определение функции, например

```
&& вам нужны только wParam и lParam в качестве параметров  
BINDEVENTSEX(yourHwnd, WM_SOMEMESSAGE, NULL, 'SomeFunc', 'wParam,  
lParam')
```

```
&& если вам вообще не нужны параметры, передайте пустую  
строку
```

```
BINDEVENTSEX(yourHwnd, WM_SOMEMESSAGE, NULL, 'SomeFunc', '')
```

Возможные значения для списка:

Значение	Описание
hWnd, uMsg, wParam или lParam	параметр как целое число со знаком
UNSIGNED(wParam или lParam)	wParam или lParam как беззнаковое целое число
HIWORD(wParam или lParam)	старшее слово (верхние 16 бит) как знаковое короткое
LOWORD(wParam или lParam)	младшее слово (нижние 16 бит) как знаковое короткое
UNSIGNED(HIWORD(wParam или lParam))	старшее слово (старшие 16 бит) как беззнаковое короткое
UNSIGNED(LOWORD(wParam или lParam))	младшее слово (нижние 16 бит) как беззнаковое короткое
BOOL(wParam или lParam)	параметр преобразован в логический, 0 = .F. все остальное = .T.

nFlags (необязательный, дополнительный)

Если параметр nFlags опущен или передан 0, по умолчанию используется BINDEVENTSEX_CALL_BEFORE.

Флаг	Описание
BINDEVENTSEX_CALL_BEFORE	Ваша функция вызывается до вызова CallWindowProc .
BINDEVENTSEX_CALL_AFTER	Ваша функция вызывается после вызова CallWindowProc . Вам необходимо вернуть числовое значение из вашей функции обратного вызова, которое используется в качестве возвращаемого значения базовой оконной процедуры.
BINDEVENTSEX_RETURN_VALUE	CallWindowProc не вызывается автоматически, вы можете либо вызвать его самостоятельно в функции обратного вызова, либо пропустить. Вам необходимо вернуть числовое значение из вашей функции обратного вызова, которое используется как возвращаемое значение базовой оконной процедуры.
BINDEVENTSEX_NO_RECURSION	То же самое, что и в BINDEVENTS _VFP.AutoYield устанавливается в .F. перед вызовом вашей функции и сбрасывается в исходное значение после этого, чтобы предотвратить рекурсию.
BINDEVENTSEX_CLASSPROC	Вместо самого окна класс окна переданного <i>nHwnd</i> подклассифицируется. Все последующие окна, созданные из класса окна, автоматически подклассифицируются. !! Вам необходимо вызвать UnbindEventsEx с третьим параметром, установленным на .T., если вы отвязываете сообщение от класса окна !!

Примечание

Флаги BINDEVENTSEX_CALL_BEFORE, BINDEVENTSEX_CALL_AFTER и BINDEVENTSEX_RETURN_VALUE являются взаимоисключающими, то есть можно указать только одно из значений, в противном случае возникнет ошибка «недопустимые параметры».

Возвращаемое значение

Указатель (числовой) на исходную оконную процедуру, возвращаемый [GetWindowLong](#) (*nHwnd* , *GWL_WNDPROC*).

Замечания

BindEventsEx ведет себя почти так же, как BINDEVENT, отличия следующие:

1. Он возвращает указатель на исходную оконную процедуру вместо не имеющего смысла значения, поэтому вам не нужно вызывать [GetWindowLong](#) (*nHwnd* , *GWL_WNDPROC*).
2. Вам не нужно привязывать сообщение к методу объекта, это также может быть публичная функция / процедура.
3. Вам не нужно вызывать [CallWindowProc](#) по умолчанию, [CallWindowProc](#) автоматически вызывается с исходными параметрами до/после обратного вызова вашей функции, это поведение контролируется с помощью параметра nFlags.
4. Вы можете точно указать, какие параметры должны быть отправлены в вашу функцию / метод обратного вызова и как они должны быть маршрутированы.
5. Вы не можете передать 0 в качестве параметра *nHwnd* , как в BINDEVENT, чтобы привязаться ко всем окнам.

Чтобы эмулировать обратные вызовы для объекта, библиотека должна сделать копию

объекта в публичную переменную. Имя переменной автоматически генерируется по следующей схеме:

```
lcVarName = "__VFP2C_WCBO" + IIF ( BITAND (nFlags,  
BINDEVENTSEX_CLASSPROC) > 0, "1", "0") + "_" + ALLTRIM ( STR  
(nHwnd)) + "_" + ALLTRIM ( STR (nMsg))
```

Этот обходной путь необходим, поскольку FoxPro LCK не предоставляет функцию API для вызова методов объекта.

Публичная переменная автоматически освобождается при отвязывании сообщения.

Хотя создается копия объекта, внутренний счетчик ссылок на объект не увеличивается, публичная копия не влияет на время жизни вашего объекта (область действия).

Единственное, что вам нужно учитывать, — это то, что ваши собственные переменные не конфликтуют с приведенной выше схемой именования, что весьма маловероятно.

Пример

Свяжите событие mousemove окна с соответствующими параметрами.

```
BINDEVENTSEX(yourHwnd, WM_MOUSEMOVE, NULL,  
'MouseMoveCallback', 'wParam, LOWORD(lParam),  
HIWORD(lParam)')  
FUNCTION MouseMoveCallback(nKeyState, nXCoord, nYCoord)  
&& обрабатывать перемещение мыши ..  
ENDFUNC
```

Привяжите активацию/деактивацию окна вашего приложения.

```
BINDEVENTSEX(_SCREEN.hWnd, WM_ACTIVATE,  
'AppFocusStateChanged', 'BOOL(wParam)')  
FUNCTION AppFocusStateChanged(bState)  
? 'Приложение ' + IIF(m.bState, 'получило фокус', ;  
'потеряло фокус')  
ENDFUNC
```

Смотрите также

Ссылки

[AsyncWaitForObject](#)
[CancelWaitForObject](#)
[CreateCallbackFunc](#)
[CreatePublicShadowObjReference](#)
[DestroyCallbackFunc](#)
[ReleasePublicShadowObjReference](#)
[UnbindEventsEx](#)

Используемые функции WinApi

[SetWindowLong](#)
[GetWindowLong](#)
[SetClassLong](#)
[GetClassLong](#)
[CallWindowProc](#)
[HeapAlloc](#)
[VirtualProtect](#)

CancelFileChange

Останавливает поток, отслеживающий изменения файлов в указанном каталоге.

`CancelFileChange (nThreadHandle)`

Параметры

nThreadHandle

Дескриптор потока, возвращенный [FindFileChange](#).

Возвращаемое значение

.T. если поток, отслеживающий каталог, был завершен, .F. в противном случае.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[SetEvent](#)

CancelRegistryChange

Останавливает поток, отслеживающий изменения указанного ключа реестра.

```
CancelRegistryChange( nThreadHandle )
```

Параметры

nThreadHandle

Дескриптор потока, возвращенный [FindRegistryChange](#).

Возвращаемое значение

.Т. если поток, отслеживающий раздел реестра, был завершен, .F. в противном случае.

Смотрите также

Ссылки

[ARegistryKeys](#)
[ARegistryValues](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[SetEvent](#)

CancelWaitForObject

Останавливает поток, ожидающий сигнала HANDLE.

```
CancelWaitForObject( nThreadHandle )
```

Параметры

nThreadHandle

Дескриптор потока, возвращенный из [AsyncWaitForObject](#).

Возвращаемое значение

.Т. если поток, отслеживающий дескриптор, был завершен, .F. в противном случае.

Смотрите также

Ссылки

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CreateCallbackFunc](#)

[CreatePublicShadowObjReference](#)

[DestroyCallbackFunc](#)

[ReleasePublicShadowObjReference](#)

[UnbindEventsEx](#)

Используемые функции WinApi

[SetEvent](#)

CenterWindowEx

Центрирует окно либо в определенном окне, либо в его родительском окне, либо на рабочем столе.

```
CenterWindowEx( hWnd , nCenterInHwnd )
```

Параметры

hWnd

Дескриптор окна, которое должно располагаться по центру.

nCenterInHwnd (необязательно)

Дескриптор окна, в котором центрируется окно.

Если опустить nCenterInHwnd, окно центрируется в родительском окне, если родительского окна нет, оно центрируется на рабочем столе.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AWindowProps](#)

[AWindows](#)

[AWindowsEx](#)

[GetWindowRectEx](#)

[GetWindowTextEx](#)

Используемые функции WinApi

[SetWindowPos](#)

[GetWindowRect](#)

[GetParent](#)

[GetSystemMetrics](#) с параметром SM_CMONITORS

[SystemParametersInfo](#) с параметром SPI_GETWORKAREA

[MonitorFromWindow](#)

ChangeSQLDataSource

Изменяет источник данных ODBC.

```
ChangeSQLDataSource( cDataSource , cDriver , nDataSourceType )
```

Параметры

cDataSource

Список атрибутов в виде пар ключевое слово-значение, разделенных [CHR\(0\)](#).
Например, "DSN=Personnel Data" + [CHR\(0\)](#) + "UID=Smith" + [CHR\(0\)](#) + "PWD=Sesame"
+ [CHR\(0\)](#) + "DATABASE=Personnel"

cDriver

Имя драйвера, список драйверов можно получить с помощью функции [ASQLDrivers](#).

nDataSourceType

Одно из следующих значений.

DatasourceType	Описание
ODBC_USER_DSN	Изменяет источники данных пользователя.
ODBC_SYSTEM_DSN	Изменяет источники данных системы.

Возвращаемое значение

.T. если вызов API к [SQLConfigDataSource](#) прошел успешно, .F. в противном случае.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#)
[SQLProceduresEx](#) [SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLConfigDataSource](#)

CloseRegistryKey

Закрывает предоставленный дескриптор ключа реестра (HKEY).

```
CloseRegistryKey( nRegKey )
```

Параметры

nRegKey

Дескриптор ключа реестра, возвращаемый [CreateRegistryKey](#) или [OpenRegistryKey](#).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegCloseKey](#)

CloseServiceHandleEx

Закрывает предоставленный дескриптор службы (SC_HANDLE).

```
CloseServiceHandleEx( nServiceHandle )
```

Параметры

nServiceHandle

Дескриптор службы Windows, полученный из [OpenServiceEx](#).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[ContinueService](#)

[ControlServiceEx](#)

[CreateServiceEx](#)

[OpenServiceEx](#)

[PauseService](#)

[StartServiceEx](#)

[StopServiceEx](#)

[WaitForServiceStatus](#)

Используемые функции WinApi

[CloseServiceHandle](#)

CLSIDFromProgID

Извлекает двоичный CLSID для указанного ProgID (имя класса COM), например «VisualFoxPro.Application».

```
CLSIDFromProgID( cProgID )
```

Параметры

cProgID

ProgID, для которого необходимо получить CLSID.

Возвращаемое значение

CLSID (двоичные символы).

Замечания

Программный идентификатор (ProgID) — это запись реестра, которая может быть связана с CLSID. Как и CLSID, ProgID идентифицирует класс, но с меньшей точностью, поскольку не гарантируется его глобальная уникальность. Формат ProgID — <Program>.<Component>.<Version>, разделенный точками и без пробелов, как в Word.Document.6.

Пример

```
lcCLSID = CLSIDFromProgID('VisualFoxPro.Application')  
&& lcCLSID теперь содержит CLSID в двоичном формате  
? StringFromCLSID (lcCLSID) && преобразовать в удобочитаемый  
формат
```

Смотрите также

Ссылки

[CLSIDFromString](#)
[CreateGuid](#)
[CreateThreadObject](#)
[GetIUnknown](#)
[IsEqualGuid](#)
[ProgIDFromCLSID](#)
[RegisterActiveObject](#)
[RegisterObjectAsFileMoniker](#)
[RevokeActiveObject](#)
[StringFromCLSID](#)

Используемые функции WinApi

[CLSIDFromProgID](#)

CLSIDFromString

Преобразует понятный человеку CLSID в двоичный CLSID. Это обратная функция [StringFromCLSID](#).

```
CLSIDFromString( cClsID )
```

Параметры

cClsId

Удобочитаемый CLSID, например «{002D2B10-C1FA-4193-B134-D86EAECC5250}».

Возвращаемое значение

CLSID (двоичный символ).

Замечания

CLSID — это глобальный уникальный идентификатор, который идентифицирует объект класса COM. Если ваш сервер или контейнер позволяет связываться со своими встроенными объектами, вам необходимо зарегистрировать CLSID для каждого поддерживаемого класса объектов.

Пример

```
lcGuidString =  
StringFromCLSID(CLSIDFROMPROGID("Word.Application"))  
lcClsId = CLSIDFromString(lcGuidString)  
? lcClsId
```

Смотрите также

Ссылки

[CLSIDFromProgID](#)
[CreateGuid](#)
[CreateThreadObject](#)
[GetIUnknown](#)
[IsEqualGuid](#)
[ProgIDFromCLSID](#)
[RegisterActiveObject](#)
[RegisterObjectAsFileMoniker](#)
[RevokeActiveObject](#)
[StringFromCLSID](#)

Используемые функции WinApi

[CLSIDFromString](#)

Colors2RGB

Преобразует предоставленные значения красного, зеленого и синего в 32-битное целое число.

```
Colors2RGB( nRed , nGreen , nBlue [, nAlpha ] )
```

Параметры

nRed

Красный компонент. 0-255

nGreen

Зеленый компонент. 0-255

nBlue

Синий компонент. 0-255

nAlpha (необязательно)

по умолчанию = 0

Альфа-канал (уровень прозрачности) цвета.

Возвращаемое значение

Комбинированное значение RGB (числовое).

Смотрите также

Ссылки

[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

CompactMem

Возвращает размер наибольшего выделенного свободного блока в куче библиотеки.

CompactMem ()

Возвращаемое значение

Возвращает размер наибольшего выделенного свободного блока в куче.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapCompact](#)

CompareFileTimes

Сравнивает время последней записи двух файлов.

```
CompareFileTimes( cFileName1 , cFileName2 )
```

Параметры

cFileName1

Полный путь к первому файлу.

cFileName2

Полный путь ко второму файлу.

Возвращаемое значение

Если время последней записи одинаково: 0

Если время последней записи первого файла больше, чем второго: 1

Если время последней записи первого файла меньше, чем второго: 2

Смотрите также

Ссылки

[ADirectoryInfo](#)

[ADirEx](#)

[ADriveInfo](#)

[AFileAttributes](#)

[AFileAttributesEx](#)

[CancelFileChange](#)

[CopyFileEx](#)

[DeleteDirectory](#)

[DeleteFileEx](#)

[FindFileChange](#)

[GetFileAttributesEx2](#)

[GetFileOwner](#)

[GetFileSizeEx2](#)

[GetFileTimes](#)

[GetLongPathNameEx](#)

[GetShortPathNameEx](#)

[MoveFileEx](#)

[SetFileAttributesEx](#)

[SetFileTimes](#)

Используемые функции WinApi

[CreateFile](#)

[GetFileTime](#)

ContinueService

Отправляет запрос на продолжение указанной службе Windows.

```
ContinueService( cServiceName | nServiceHandle [, nTimeout [,  
cServer [, cDatabase ]]])
```

Параметры

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

nTimeout (необязательно)

Максимальное время ожидания в секундах, пока служба находится в состоянии SERVICE_START_PENDING.

Если вы передадите 0 в качестве тайм-аута, функция не будет ждать, пока служба будет инициализирована, а вместо этого она немедленно вернет управление после отправки запроса на запуск.

Если вы опустите этот параметр или передадите NULL, тайм-аут будет установлен на тайм-аут по умолчанию, сообщенный службой. См. справку MSDN для члена "dwWaitHint" структуры [SERVICE_STATUS_PROCESS](#).

cServer (необязательно)

Имя сервера, на котором запущена служба.
См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.
См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

Если служба уже находится в запущенном состоянии или продолжает работу в течение указанного периода ожидания, возвращается 1, если время ожидания истекло, возвращается 0.

Смотрите также

Ссылки

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[ControlService](#)
[QueryServiceStatus](#)
[OpenSCManager](#)
[OpenService](#)
[CloseServiceHandle](#)

ControlServiceEx

Отправляет пользовательский запрос управления указанной службе Windows.

```
ControlServiceEx( cServiceName | nServiceHandle , nControlCode [,  
cServer [, cDatabase ]])
```

Параметры

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

nControlCode

Пользовательский код управления для отправки в службу.

Допустимые коды управления находятся в диапазоне от 128 до 255.

cServer (необязательно)

Имя сервера, на котором запущена служба.

См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.

См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[ControlService](#)
[OpenSCManager](#)
[OpenService](#)
[CloseServiceHandle](#)

CopyFileEx

Копирует файл, может быть передана дополнительная функция обратного вызова, которая получает статус во время выполнения операции копирования.

```
CopyFileEx( cSourceFile , cDestinationFile [, cCallback [, nFlags  
[, nCallbackLimiter ]]])
```

Параметры

cSourceFile

Полный путь к файлу для копирования.

cDestinationFile

Полностью определенный путь к месту назначения для операции копирования.

cCallback (необязательно)

Функция обратного вызова с информацией о ходе выполнения во время операции копирования.

Функция должна иметь следующий прототип:

```
FUNCTION CopyCallback(nBytesCopied, nFileSize,  
nPercentCopied)  
ENDFUNC
```

Примечание

Из функции можно вернуть следующие значения:

PROGRESS_CONTINUE - продолжает операцию копирования

PROGRESS_CANCEL - отменяет операцию копирования

PROGRESS_QUIET - продолжает операцию копирования, но останавливает все дальнейшие обратные вызовы

PROGRESS_STOP - останавливает операцию копирования

.T. - равно PROGRESS_CONTINUE

.F. - равно PROGRESS_CANCEL

nFlags (необязательно)

по умолчанию COPY_FILE_NO_BUFFERING

Возможные значения и их значение см. в оригинальной документации [CopyFileEx](#) (параметр dwCopyFlags).

nCallbackLimiter (необязательно)

Время в миллисекундах, которое управляет частотой вызова функции обратного вызова.

По умолчанию 100 миллисекунд.

Установка более низких значений может снизить пропускную способность копирования.

Возвращаемое значение

Состояние операции копирования.

PROGRESS_CONTINUE (0), PROGRESS_CANCEL (1), PROGRESS_STOP (2) или PROGRESS_QUIET (3).

PROGRESS_CONTINUE и PROGRESS_QUIET означают успешное выполнение.

PROGRESS_STOP означает, что операция копирования была остановлена, а целевой файл остался на диске.

PROGRESS_CANCEL означает, что операция копирования была прервана, а целевой файл удален.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[CopyFileEx](#)

CreateCallbackFunc

Создает ассемблерный преобразователь, который эмулирует функцию обратного вызова C.

```
CreateCallbackFunc( cCallback , cReturnType , cParamTypes [,
oObject [, nFlags ]])
```

Параметры

cCallback

Имя функции для обратного вызова.

cReturnType

Тип данных возвращаемого значения функции обратного вызова C.

Один из:

Тип данных	Описание
VOID или empty string	нет возвращаемого значения
INTEGER или LONG	знаковое 32-битное целое число
UINTeger или ULONG	беззнаковое 32-битное целое число
SHORT	знаковое 16-битное целое число
USHORT	беззнаковое 16-битное целое число
BOOL	логическое 32-битное значение - 0 отображается в .F. все остальное - в .T.
SINGLE	32-битное значение с плавающей точкой
DOUBLE	64-битное значение с плавающей точкой

cParamTypes

Список типов параметров, разделенных запятыми, которые ожидает функция C.

Допустимые типы:

Тип данных	Описание
INTEGER или LONG	знаковое 32-битное целое число - в 64-битной версии библиотеки LONG является синонимом INT64
UINTeger или ULONG	беззнаковое 32-битное целое число
SHORT	знаковое 16-битное целое число
USHORT	беззнаковое 16-битное целое число
BOOL	логическое 32-битное значение - 0 отображается в .F., все остальное - в .T.
SINGLE (0-6)	32-битное значение с плавающей точкой *
DOUBLE (0-16)	64-битное значение с плавающей точкой *
STRING (ANSI UNICODE)	строка в стиле C **
INT64 (BINARY LITERAL CURRENCY)	знаковое 64-битное целое число ***
UINT64 (BINARY LITERAL CURRENCY)	беззнаковое 64-битное целое число ***

Примечание

* Для типов данных SINGLE и DOUBLE можно указать необязательное значение точности, например:
 InCallback = CreateCallbackFunc('Foo','DOUBLE 15','INTEGER, INTEGER')
 Если вы не укажете значение точности, будет использовано значение по умолчанию 6.

Примечание

** По умолчанию строки в стиле C маршируются как числовые значения указателей. Вы можете получить фактическую строку, передав значение функции [ReadCString](#) (Ansi) или [ReadWString](#) (Unicode). Если вы дополнительно передаете ANSI или UNICODE, строка маршируется этими функциями автоматически.

Примечание

*** По умолчанию 64-битные целые числа со знаком и без знака маршируются как числовые значения. Обратный вызов усечет значения, если параметр превышает числовой предел VFP. Если вы не уверены на 100%, что такие большие числа никогда не передаются, укажите модификатор BINARY, LITERAL или CURRENCY. BINARY марширует параметр как 8-байтовое значение varbinary, LITERAL марширует параметр как строку цифр, а CURRENCY как литерал валюты.

Примечание

Лучше всего использовать LONG для любого типа данных указателя, это позволит маршировать правильные значения в 32- или 64-битной версии библиотеки соответственно. Количество параметров ограничено 27, пределом параметров, которые может исключить функция VFP.

oObjectRef (необязательно)

Если метод FoxPro должен быть вызван обратно по ссылке на объект, вы можете передать ссылку на объект в этом параметре. Если вы хотите передать параметр nFlags, но не хотите передавать объект, передайте вместо этого NULL.

nFlags (необязательный, дополнительный)

по умолчанию = CALLBACK_SYNCHRONOUS

допустимые значения:

Флаг	Описание
CALLBACK_SYNCHRONOUS	Функция обратного вызова C вызывается в основном потоке Visual FoxPro. Для всех функций Winapi, например EnumWindows .
CALLBACK_ASYNCHRONOUS_POST	Функция обратного вызова C вызывается в отдельном потоке для сторонних C DLL, которые вызывают события (вызывают функцию C). Сам обратный вызов отправляется асинхронно в основной поток Visual FoxPro, он не блокирует сторонний поток, событие обрабатывается, когда FoxPro находится в состоянии READ EVENTS.
CALLBACK_ASYNCHRONOUS_SEND	Функция обратного вызова C вызывается в отдельном потоке для сторонних C DLL, которые вызывают события (вызывают функцию C). Обратный вызов отправляется синхронно в основной поток Visual FoxPro, поток возобновляется после возврата функции FoxPro.
CALLBACK_CDECL	Функция C использует соглашение о вызовах <i>cdecl</i> , по умолчанию функция обратного вызова создается с

соглашением о вызовах *stdcall* .

Примечание

Если указать CALLBACK_ASYNCRONOUS_POST, несколько обратных вызовов могут перекрывать друг друга.

Чтобы обойти это, установите _VFP.AutoYield в .F. — либо для всего приложения, либо внутри функции FoxPro, которая используется как функция обратного вызова, например

```
FUNCTION YourCallbackFunction(lnParameter1, lnParameter2)
  _VFP.AutoYield = .F.
  && делай свое дело...
  _VFP.AutoYield = .T.
ENDFUNC
```

Возвращаемое значение

Указатель (числовой) на созданную функцию обратного вызова.

Замечания

Для эмуляции обратных вызовов на объекте библиотека должна сделать копию объекта в публичную переменную.

Имя переменной автоматически генерируется по следующей схеме:

```
lcVarName = "__VFP2C_CBO_" + ALLTRIM(STR(returnvalue))
```

Этот обходной путь необходим, поскольку FoxPro LCK не предоставляет функцию API для вызова методов объекта.

Публичная переменная автоматически освобождается при отвязывании сообщения.

Хотя создается копия объекта, внутренний счетчик ссылок на объект не увеличивается, публичная копия не влияет на время жизни вашего объекта (область действия).

Единственное, что вам нужно учитывать, — это то, что ваши собственные переменные не конфликтуют с приведенной выше схемой именования, что весьма маловероятно.

Смотрите также

Ссылки

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreatePublicShadowObjReference](#)

[DestroyCallbackFunc](#)

[ReleasePublicShadowObjReference](#)

[UnbindEventsEx](#)

CreateGuid

Создает новый GUID (глобальный уникальный идентификатор).

```
CreateGuid([ nOutputFormat ])
```

Параметры

nOutputFormat (необязательно)

по умолчанию = CREATE_GUID_ANSI

Одно из следующих значений.

Формат вывода	Возвращаемое значение
CREATE_GUID_ANSI	Guid возвращается как строка ANSI
CREATE_GUID_UNICODE	Guid возвращается как строка Unicode
CREATE_GUID_BINARY	Guid возвращается в двоичном формате

Возвращаемое значение

GUID, формат которого зависит от отправляемого вами параметра.

Замечания

Хотя GUID обычно считается глобально уникальным, нет никаких гарантий, что он является универсально уникальным. Следует соблюдать осторожность, если вы собираетесь развернуть свое приложение на других планетах.

Если вы собираетесь использовать эту функцию для создания GUID в качестве первичного ключа для таблицы VFP с использованием "Значения по умолчанию" таблицы, эта функция должна быть в области действия при добавлении новых записей, иначе вы получите ошибку. Вы можете использовать триггеры базы данных для загрузки VFP2C32 при открытии базы данных, чтобы гарантировать это.

Пример

```
cId = CreateGuid()  
? cId  
* Возвращает понятный человеку 36-символьный GUID
```

Смотрите также

Ссылки

[CLSIDFromProgID](#)
[CLSIDFromString](#)
[CreateThreadObject](#)
[GetIUnknown](#)
[IsEqualGuid](#)
[ProgIDFromCLSID](#)
[RegisterActiveObject](#)
[RegisterObjectAsFileMoniker](#)
[RevokeActiveObject](#)
[StringFromCLSID](#)

Используемые функции WinApi

[CoCreateGuid](#)
[StringFromGUID2](#)

CreatePublicShadowObjReference

Создает новую публичную переменную, ссылающуюся на предоставленный объект, не увеличивая счетчик ссылок на объекты.

```
CreatePublicShadowObjReference( cVariableName , oObject )
```

Параметры

cVariableName

Имя публичной переменной, которую следует создать.

oObject

Объект, на который должна ссылаться публичная переменная.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreateCallbackFunc](#)

[DestroyCallbackFunc](#)

[ReleasePublicShadowObjReference](#)

[UnbindEventsEx](#)

CreateRegistryKey

Создает или открывает раздел реестра.

```
CreateRegistryKey( nRegKey , cKeyName [, nAccessRights [, nOptions  
[, cClass ]]])
```

Параметры

nRegKey

Либо дескриптор реестра, возвращенный [OpenRegistryKey](#), либо одна из следующих констант ключа.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и</p>

	<p>устанавливается при первой ссылке процесса на HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>
HKEY_PERFORMANCE_NLSTEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe.</p> <p>Windows 2000: этот ключ не поддерживается.</p>
HKEY_PERFORMANCE_TEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на</p>

	американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe. Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

cKeyName

Имя подключаемого ключа, который должен быть создан.

Если *cKeyName* — пустая строка, функция возвращает новый дескриптор ключа, указанного *nRegKey*.

nAccessRights (необязательно)

по умолчанию = KEY_ALL_ACCESS

Одно или комбинация следующих значений.

Константа	Описание
KEY_ALL_ACCESS	Объединяет права доступа STANDARD_RIGHTS_REQUIRED, KEY_QUERY_VALUE, KEY_SET_VALUE, KEY_CREATE_SUB_KEY, KEY_ENUMERATE_SUB_KEYS, KEY_NOTIFY и KEY_CREATE_LINK.
KEY_CREATE_LINK	Зарезервировано для использования системой.
KEY_CREATE_SUB_KEY	Требуется для создания подраздела ключа реестра.
KEY_ENUMERATE_SUB_KEYS	Требуется для перечисления подразделов ключа реестра.
KEY_EXECUTE	Эквивалентно KEY_READ.
KEY_NOTIFY	Требуется для запроса уведомлений об изменениях для раздела реестра или для подразделов раздела реестра.
KEY_QUERY_VALUE	Требуется для запроса значений ключа реестра.
KEY_READ	Объединяет значения STANDARD_RIGHTS_READ, KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS и KEY_NOTIFY.
KEY_SET_VALUE	Требуется для создания, удаления или установки значения реестра.
KEY_WOW64_32KEY	Указывает, что приложение в 64-разрядной версии Windows должно работать в 32-разрядном представлении реестра. Для получения дополнительной информации см. Доступ к альтернативному представлению реестра. Этот флаг должен быть объединен с помощью оператора OR с другими флагами в этой таблице, которые либо запрашивают, либо получают доступ к значениям реестра. Windows 2000: Этот флаг не поддерживается.
KEY_WOW64_64KEY	Указывает, что приложение в 64-разрядной версии Windows должно работать в 64-разрядном представлении реестра. Для получения дополнительной информации см. Доступ к альтернативному представлению реестра. Этот флаг должен быть объединен с помощью

	оператора OR с другими флагами в этой таблице, которые либо запрашивают, либо получают доступ к значениям реестра. Windows 2000: Этот флаг не поддерживается.
KEY_WRITE	Объединяет права доступа STANDARD_RIGHTS_WRITE, KEY_SET_VALUE и KEY_CREATE_SUB_KEY.

Примечание

Для получения дополнительной информации ознакомьтесь с разделом [Безопасность ключей реестра и права доступа](#).

nOptions (необязательно)

по умолчанию = REG_OPTION_NON_VOLATILE

Одно из следующих значений:

Значение	Описание
REG_OPTION_BACKUP_RESTORE	Если этот флаг установлен, функция игнорирует параметр samDesired и пытается открыть ключ с правами доступа, необходимыми для резервного копирования или восстановления ключа. Если вызывающий поток имеет включенную привилегию SE_BACKUP_NAME, ключ открывается с правами доступа ACCESS_SYSTEM_SECURITY и KEY_READ. Если вызывающий поток имеет включенную привилегию SE_RESTORE_NAME, ключ открывается с правами доступа ACCESS_SYSTEM_SECURITY и KEY_WRITE. Если включены обе привилегии, ключ имеет объединенные права доступа для обеих привилегий. Для получения дополнительной информации см. Запуск со специальными привилегиями.
REG_OPTION_CREATE_LINK	Этот ключ является символической ссылкой. Целевой путь назначается значению L"SymbolicLinkValue" ключа. Целевой путь должен быть абсолютным путем реестра. Символические ссылки реестра следует использовать только в случае крайней необходимости для совместимости приложений.
REG_OPTION_NON_VOLATILE	Этот ключ не является энергозависимым; это значение по умолчанию. Информация хранится в файле и сохраняется при перезапуске системы. Функция RegSaveKey сохраняет ключи, которые не являются энергозависимыми.
REG_OPTION_VOLATILE	Все ключи, созданные функцией, являются изменчивыми. Информация хранится в памяти и не сохраняется при выгрузке соответствующего куста реестра. Для HKEY_LOCAL_MACHINE это происходит при завершении работы системы. Для ключей реестра, загруженных функцией RegLoadKey, это происходит при выполнении соответствующего RegUnLoadKey. Функция RegSaveKey не сохраняет изменчивые ключи. Этот флаг игнорируется для ключей, которые уже существуют.

cClass (необязательно)

по умолчанию = NULL

Пользовательский тип класса этого ключа. Этот параметр можно игнорировать.

Возвращаемое значение

Дескриптор (числовой) ключа реестра.

Замечания

Функция `CreateRegistryKey` создает все отсутствующие ключи в указанном пути. Приложение может воспользоваться этим поведением, чтобы создать несколько ключей одновременно. Например, приложение может создать подключ на четыре уровня в глубину одновременно с тремя предыдущими подключами, указав строку следующего вида для параметра `lpSubKey`: "subkey1\subkey2\subkey3\subkey4"

Обратите внимание, что это поведение приведет к созданию нежелательных ключей, если существующий ключ в пути написан неправильно.

Приложение не может создать ключ, который является прямым потомком `HKEY_USERS` или `HKEY_LOCAL_MACHINE`, но оно может создавать подключи на более низких уровнях деревьев `HKEY_USERS` или `HKEY_LOCAL_MACHINE`.

Смотрите также

Ссылки

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegCreateKeyEx](#)

CreateServiceEx

Устанавливает службу Windows.

```
CreateServiceEx( cServiceName , cDisplayName , cExecutable [,
nServiceType [, nStartType [, nErrorControl [, cLoadOrderGroup [,
cDependencies [, cServiceAccount [, cAccountPassword [, cMachine [,
cDatabase ]]]]]]]])
```

Параметры

cServiceName

Краткое название службы.

cDisplayName

Описательное название службы.

cExecutable

Полный путь к исполняемому файлу службы.

nServiceType (необязательно)

по умолчанию = SERVICE_WIN32_OWN_PROCESS

Тип службы.

Этот параметр может иметь одно из следующих значений.

Тип службы	Описание
SERVICE_FILE_SYSTEM_DRIVER	Служба драйвера файловой системы.
SERVICE_KERNEL_DRIVER	Служба драйвера.
SERVICE_WIN32_OWN_PROCESS	Служба, работающая в собственном процессе.
SERVICE_WIN32_SHARE_PROCESS	Служба, которая разделяет процесс с одной или несколькими другими службами.
SERVICE_INTERACTIVE_PROCESS	Если вы указали SERVICE_WIN32_OWN_PROCESS или SERVICE_WIN32_SHARE_PROCESS, и служба работает в контексте учетной записи LocalSystem, вы также можете указать это значение.

nStartType (необязательно)

по умолчанию = SERVICE_AUTO_START

Параметры запуска службы.

Этот параметр может иметь одно из следующих значений.

Тип запуска	Описание
SERVICE_AUTO_START	Служба, автоматически запускаемая диспетчером управления службами во время запуска системы.
SERVICE_BOOT_START	Драйвер устройства, запущенный загрузчиком системы. Это значение действительно только для служб драйверов.
SERVICE_DEMAND_START	Служба, запускаемая диспетчером управления службами, когда процесс вызывает функцию StartService .
SERVICE_DISABLED	Служба, которая не может быть запущена. Попытки запустить службу приводят к коду ошибки ERROR_SERVICE_DISABLED.

SERVICE_SYSTEM_START	Драйвер устройства, запущенный функцией IoInitSystem. Это значение действительно только для служб драйвера.
----------------------	---

nErrorControl

по умолчанию = SERVICE_ERROR_NORMAL

Серьезность ошибки и предпринимаемые действия, если эта служба не запускается.

Этот параметр может иметь одно из следующих значений.

Контроль ошибок	Описание
SERVICE_ERROR_CRITICAL	Программа запуска регистрирует ошибку в журнале событий, если это возможно. Если запускается последняя известная удачная конфигурация, операция запуска завершается неудачей. В противном случае система перезапускается с последней известной удачной конфигурацией.
SERVICE_ERROR_IGNORE	Программа запуска игнорирует ошибку и продолжает операцию запуска.
SERVICE_ERROR_NORMAL	Программа запуска регистрирует ошибку в журнале событий, но продолжает операцию запуска.
SERVICE_ERROR_SEVERE	Программа запуска регистрирует ошибку в журнале событий. Если запускается последняя удачная конфигурация, операция запуска продолжается. В противном случае система перезапускается с последней удачной конфигурацией.

cLoadOrderGroup (необязательно)

по умолчанию = .NULL.

cDependencies (необязательно)***cServiceAccount (необязательно)******cAccountPassword (необязательно)******cMachine (необязательно)******cDatabase (необязательно)*****Возвращаемое значение**

Всегда .T.

Смотрите также**Ссылки**

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[OpenSCManager](#)[CreateService](#)[CloseServiceHandle](#)

CreateSQLDataSource

Создает источник данных ODBC.

```
CreateSQLDataSource( cDataSource , cDriver , nDataSourceType )
```

Параметры

cDataSource

Список атрибутов в виде пар ключевое слово-значение, разделенных [CHR](#)(0).
Например, "DSN=Personnel Data" + [CHR](#)(0) + "UID=Smith" + [CHR](#)(0) + "PWD=Sesame"
+ [CHR](#)(0) + "DATABASE=Personnel"

cDriver

Имя драйвера, список драйверов можно получить с помощью функции [ASQLDrivers](#).

nDataSourceType

Одно из следующих значений.

Тип источника данных	Описание
ODBC_USER_DSN	Создает пользовательские источники данных.
ODBC_SYSTEM_DSN	Создает системные источники данных.

Возвращаемое значение

.T. если источник данных был создан, .F. в противном случае.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLConfigDataSource](#)

CreateThreadObject

Создает COM-объект в отдельном потоке.

```
CreateThreadObject( cClassName [, oCallback [, bSynchronousAccess [,
nContext [, nStackSize ]]])
```

Параметры

cClassName

Указывает класс COM, из которого создается новый объект.

oCallback

по умолчанию = .NULL.

Ссылка на объект, которая будет получать обратные вызовы, когда вызываемые методы созданного объекта завершатся или произойдет ошибка. Если вы передадите .NULL., вы не получите никаких обратных вызовов при завершении метода или ошибках. Если вы передадите объект, он должен реализовать по крайней мере один из этих методов:

```
FUNCTION OnCallComplete(callid, result, callcontext)
&& callid = уникальный идентификатор вызова (целое число),
&&          который назначается каждому вызову
&& result = возвращаемое значение метода
&& callcontext = значение callcontext, связанное с вызовом

FUNCTION OnError(callid, callcontext, errornumber, ;
    errorsource, errordescription)
&& callid = уникальный идентификатор вызова (целое число),
&&          который назначается каждому вызову
&& callcontext = значение callcontext, связанное с вызовом
&& errornumber = номер_ошибки
&& errorsource = источник возникшей ошибки
&& errordescription = описание возникшей ошибки
```

Примечание

Вам следует реализовать правильную обработку ошибок COM, используя функцию [COMRETURNERROR](#) внутри ваших классов COM, чтобы обратный вызов *OnError* возвращал корректную информацию.

bSynchronousAccess

по умолчанию = .F.

Если .T. созданный прокси-объект будет поддерживать свойство «Объект» для синхронного доступа к созданному объекту.

nContext

по умолчанию = CLSCTX_INPROC_SERVER

Контекст выполнения созданного COM-объекта.

Этот параметр может иметь одно из следующих значений.

Контекст исполнения	Описание
CLSCTX_INPROC_SERVER	Код, который создает и управляет объектами этого класса, представляет собой DLL, которая выполняется в том же процессе, что и вызывающая функция, указывающая контекст класса.
CLSCTX_LOCAL_SERVER	Код EXE, который создает и управляет объектами этого

класса, выполняется на той же машине, но загружается в отдельное пространство процесса.

nStackSize

по умолчанию = 64 KB

Указывает размер стека созданного потока в байтах.

Возвращаемое значение

Прокси-объект для созданного объекта, позволяющий вызывать функции асинхронно.

Замечания

Эта функция позволяет создавать настоящие многопоточные программы FoxPro. Создание многопоточного кода в FoxPro стало возможным, поскольку можно было скомпилировать код FoxPro в классы COM. Проблема заключалась в том, что этот код можно было запустить в многопоточном режиме только в других средах/языках, поддерживающих многопоточность, таких как IIS, C++ или C#.

Функция *CreateThreadObject* обеспечивает недостающую часть головоломки — создание многопоточных объектов COM в их собственном потоке из Visual FoxPro.

Каждый вызов метода или доступ к свойству/присвоению, которые вы делаете, выполняется в потоке, в котором «живет» созданный объект, вызов немедленно возвращается, пока метод выполняется в фоновом режиме. Возвращаемое значение всегда является уникальным значением идентификатора (callid), назначенным вызову.

Все вызовы помещаются в очередь FIFO (первым пришел — первым обслужен) и обрабатываются в порядке их выполнения. Размер очереди ограничен только доступной памятью, поэтому вы можете ставить в очередь столько вызовов объекта, сколько захотите.

Время жизни потока привязано к возвращаемому прокси-объекту. Если прокси-объект уничтожен, поток также уничтожается. Если поток все еще выполняет метод, когда объект уничтожен, уничтожение будет заблокировано до тех пор, пока метод не завершит выполнение. Все остальные вызовы, которые все еще находятся в очереди, отбрасываются.

Возвращаемый прокси-объект также реализует несколько функций для управления прерыванием метода и другими задачами.

Тип	Имя	Описание
Method	AbortCall(callid)	Прерывает вызов с указанным идентификатором вызова или все вызовы, если передан 0.
Method	GetCallQueueSize	Возвращает количество ожидающих вызовов в очереди. Текущий обрабатываемый вызов также учитывается.
Method	GetCallRunTime(callid)	Возвращает время выполнения вызова в миллисекундах. Возвращается 0, если вызов завершен или все еще находится в очереди. Если вы передаете 0, возвращается время выполнения текущего обрабатываемого вызова.
Property	Object	При доступе к свойству <i>Object</i> все вызовы производятся синхронно. Это свойство доступно только при передаче .T. в параметре <i>bSynchronousAccess</i> при создании объекта.
Property	ThreadId	Свойство только для чтения, возвращающее

		идентификатор потока, в котором работает объект.
Property/Method	CallContext	Свойство/метод <i>CallContext</i> позволяет связать значение (например, первичный ключ строки таблицы) со следующим асинхронным вызовом метода. Это значение передается обратному вызову завершения, когда метод завершается.

Если вы реализуете свойство с именем *CallInfo* в классе COM, созданном с помощью *CreateThreadObject*, вы можете получить доступ к этому свойству во время выполнения методов для проверки прерывания метода и другой информации. Свойство вернет объект, который реализует следующий интерфейс.

Тип	Имя	Описание
Property	AbortEvent	Возвращает объект события WinAPI (CreateEvent), который используется внутренне для прерывания вызова.
Method	Aborted()	Возвращает .T., если вызов был прерван, .F. в противном случае.
Property	CallContext	Возвращает значение, связанное с вызовом метода.
Property	CallId	Возвращает уникальный идентификатор, назначенный этому вызову.

Пример

TestFunc блокирует переданное время ожидания.

TestFunc2 циклирует, имитирует некоторую работу с вызовом API Sleep и проверяет прерывание метода.

```
&& Скомпилировать этот класс в «Многопоточный COM-сервер
(dll)»
DEFINE CLASS ExampleObject AS Session OLEPUBLIC
    DataSession = 2 && вы всегда должны запускать каждый
                        && объект в его собственном сеансе данных
    CallInfo = .NULL. && "магическое" свойство

    FUNCTION TestFunc(lnTimeOut AS Long) AS Long
        DECLARE Sleep IN WIN32API INTEGER
        Sleep(m.lnTimeOut * 1000)
        RETURN m.lnTimeOut
    ENDFUNC

    FUNCTION TestFunc2() AS Long
        LOCAL lnRetVal, xj
        DECLARE Sleep IN WIN32API INTEGER
        m.lnRetVal = 1
        FOR m.xj = 1 TO 30
            Sleep(500)
            IF THIS.CallInfo.Aborted()
                m.lnRetVal = -1
                EXIT
            ENDIF
        ENDFOR
        RETURN m.lnRetVal
```

```

        ENDFUNC

    ENDDDEFINE

    && используя указанный выше класс
    PUBLIC loObj, loCallback, xj, lnCallId
    m.loCallback = CREATEOBJECT('ExampleCallback')
    m.loObj = CreateThreadObject('YourProject.ExampleObject', ;
        m.loCallback, .T.)

    FOR m.xj = 1 to 5
        ? m.loObj.TestFunc(m.xj)
    ENDFOR

    && прерывание вызова
    m.lnCallId = m.loObj.TestFunc2()
    ? m.loObj.AbortCall(m.lnCallId)

    && связывание значения со следующим вызовом
    m.loObj.CallContext = 'someValue'
    ? m.loObj.TestFunc(5)
    && Вы также можете использовать CallContext как метод с
    && цепочкой (он возвращает «THIS»)
    ? m.loObj.CallContext('someValue2').TestFunc(5)

    && выполнение синхронного вызова по свойству "Object".
    && это заблокирует до тех пор, пока метод не завершится,
    && как и любой обычный вызов метода
    ? m.loObj.Object.TestFunc(5)

    DEFINE CLASS ExampleCallback AS Custom

        FUNCTION OnCallComplete(callid AS Long, ;
            result AS Variant, ;
            callcontext AS Variant) AS VOID
            ? callid, result, callcontext
        ENDFUNC

        FUNCTION OnError(callid AS Long, ;
            callcontext AS Variant, ;
            errornumber AS Long, errorsource AS String, ;
            errordescription AS String) AS VOID
            ? callid, callcontext, errornumber, errorsource, ;
            errordescription
        ENDFUNC

    ENDDDEFINE

```

Смотрите также

Ссылки

[CLSIDFromProgID](#)
[CLSIDFromString](#)
[CreateGuid](#)

[GetIUnknown](#)
[IsEqualGuid](#)
[ProgIDFromCLSID](#)
[RegisterActiveObject](#)
[RegisterObjectAsFileMoniker](#)
[RevokeActiveObject](#)
[StringFromCLSID](#)

Используемые функции WinApi

[_beginthreadex](#)
[CloseHandle](#)
[CLSIDFromProgID](#)
[CoCreateInstance](#)
[CoGetInterfaceAndReleaseStream](#)
[CoMarshalInterThreadInterfaceInStream](#)
[CoInitializeEx](#)
[CoUninitialize](#)
[CreateEvent](#)
[DispatchMessage](#)
[EnterCriticalSection](#)
[GetLastError](#)
[InitializeCriticalSectionAndSpinCount](#)
[LeaveCriticalSection](#)
[MsgWaitForMultipleObjects](#)
[PeekMessage](#)
[ResetEvent](#)
[SetEvent](#)
[VariantChangeType](#)
[VariantClear](#)
[VariantInit](#)
[VariantCopy](#)
[WaitForSingleObject](#)

Decimals

Возвращает количество десятичных знаков числового значения.

```
Decimals( nValue )
```

Параметры

nValue

Числовое значение.

Возвращаемое значение

Возвращает количество цифр после десятичной точки.

Смотрите также

Ссылки

[AFontInfo](#)

[ASplitStr](#)

[GetCursorPosEx](#)

[Int64_Add](#)

[Int64_Div](#)

[Int64_Mod](#)

[Int64_Mul](#)

[Int64_Sub](#)

DeleteDirectory

Удаляет каталог, включая все файлы и подкаталоги.

```
DeleteDirectory( cDirectory )
```

Параметры

cDirectory

Полный путь к каталогу, который следует удалить.

Возвращаемое значение

Всегда .T.

Замечания

Примечание

Эта функция рекурсивно удаляет **ВСЬ** каталог, включая все файлы и подкаталоги.
Будьте осторожны при ее использовании!

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[RemoveDirectory](#)
[DeleteFile](#)
[FindFirstFile](#)
[FindNextFile](#)
[FindClose](#)

DeleteFileEx

Удаляет файл.

```
DeleteFileEx( cFilename [, nFlags ] )
```

Параметры

cFileName

Полный путь к файлу.

nFlags (необязательно)

по умолчанию = 0

Управляет поведением в случае ошибки и типом возвращаемого функцией значения.

Флаг	Описание
0	Вызывает ошибку, если она возникает. Всегда возвращает .Т.
1	Не вызывает ошибку. Функция возвращает .Т. или .F., чтобы указать, было ли удаление успешным.
2	Не вызывает ошибку. Функция возвращает 0 или числовое значение ошибки API, чтобы указать, было ли удаление успешным.
3	Не вызывает ошибку. Функция возвращает "" или сообщение об ошибке API, чтобы указать, было ли удаление успешным.

В случае возникновения ошибки дополнительная информация доступна через AERROREX.

Возвращаемое значение

Всегда .Т.

Замечания

Функция удаляет атрибут «только для чтения», который предотвращает предварительное удаление с помощью команды VFP [DELETE FILE](#), если это необходимо.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi[DeleteFile](#)[GetFileAttributes](#)[SetFileAttributes](#)

DeleteRegistryKey

Удаляет указанный раздел реестра.

```
DeleteRegistryKey( nRegKey , cKeyName [, nDeleteFunc ] )
```

Параметры

nRegKey

Либо дескриптор реестра, возвращаемый [CreateRegistryKey](#), [OpenRegistryKey](#), либо одна из следующих констант ключей:

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и устанавливается при первой ссылке процесса на</p>

	<p>HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>
HKEY_PERFORMANCE_NLSTEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe.</p> <p>Windows 2000: этот ключ не поддерживается.</p>
HKEY_PERFORMANCE_TEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на</p>

	американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe.
	Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

cKeyName

Имя ключа, который нужно удалить. Это должен быть подключ ключа, который идентифицирует *nRegKey*.

Если *nDeleteFunc* — REG_DELETE_NORMAL, ключ не может иметь подключей.

Если *nDeleteFunc* — REG_DELETE_SHELL, функция удаляет подключ и всех его потомков.

Примечание

Названия ключей не чувствительны к регистру.

nDeleteFunc (необязательно)

по умолчанию = REG_DELETE_NORMAL

Указывает функцию API для использования.

Одно из следующих значений.

Значение	Описание
REG_DELETE_NORMAL	Функция удаляет раздел реестра путем вызова RegDeleteKey .
REG_DELETE_SHELL	Функция удаляет раздел реестра путем вызова SHDeleteKey .

Возвращаемое значение

.Т. если раздел реестра был удален, .F. в противном случае.

Смотрите также**Ссылки**

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegDeleteKey](#)
[SHDeleteKey](#)

DeleteSQLDataSource

Удаляет источник данных ODBC.

```
DeleteSQLDataSource( cDataSource , cDriver , nDataSourceType )
```

Параметры

cDataSource

Имя источника данных.

cDriver

Имя драйвера, список драйверов можно получить с помощью функции [ASQLDrivers](#).

nDataSourceType

Одно из следующих значений.

Тип источника данных	Описание
ODBC_USER_DSN	Удаляет источники данных пользователя.
ODBC_SYSTEM_DSN	Удаляет системные источники данных.

Возвращаемое значение

.T. если источник данных SQL был удален, .F. в противном случае.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLConfigDataSource](#)

DestroyCallbackFunc

Освобождает переданную функцию обратного вызова C.

```
DestroyCallbackFunc( nCallbackAddress )
```

Параметры

nCallbackAddress

Указатель на функцию обратного вызова C, возвращаемую функцией [CreateCallbackFunc](#).

Возвращаемое значение

.T. если функция обратного вызова была освобождена, .F. если она не была найдена.

Смотрите также

Ссылки

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreateCallbackFunc](#)

[CreatePublicShadowObjReference](#)

[ReleasePublicShadowObjReference](#)

[UnbindEventsEx](#)

Double2DT

Преобразует числа двойной точности в дату и время.

```
Double2DT( nValue )
```

Параметры

nValue

Числовое значение.

Возвращаемое значение

Значение даты и времени, представленное переданным значением double.

Смотрите также

Ссылки

[ATimeZones](#)

[DT2Double](#)

[DT2FI](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

Double2Str

Преобразует двойное (64-битное числовое) значение в двоичную строку.

```
Double2Str( nValue )
```

Параметры

nValue

Числовое значение в диапазоне 1,7E +/- 308 (15 цифр).

Возвращаемое значение

Строка, которая в двоичном виде равна переданному значению double.

Смотрите также

Ссылки

[Colors2RGB](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

DT2Double

Преобразует значение даты и времени в число двойной точности.

```
DT2Double( tTime )
```

Параметры

tTime

Значение даты и времени.

Возвращаемое значение

Число двойной точности, представляющее переданное значение Datetime.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2FI](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

DT2FT

Преобразует значение datetime в структуру [FILETIME](#).

```
DT2FT( tTime , nFileTimePointer [, bIsUTC ])
```

Параметры

tTime

Значение даты и времени.

nFileTimePointer

Указатель на структуру [FILETIME](#).

bIsUTC (необязательно)

по умолчанию = .F.

Если .T., то предполагается, что переданная дата и время уже соответствуют времени UTC.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

Используемые функции WinApi

[LocalFileTimeToFileTime](#)

DT2ST

Преобразует значение datetime в структуру [SYSTEMTIME](#).

```
DT2ST( tTime , nSystemTimePointer [, bToUTC ] )
```

Параметры

tTime

Значение даты и времени.

nSystemTimePointer

Указатель на структуру [SYSTEMTIME](#).

bToUTC (необязательно)

по умолчанию = .F.

Если .T., то дата и время преобразуются в UTC.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

Используемые функции WinApi

[FileTimeToSystemTime](#)

[LocalFileTimeToFileTime](#)

DT2Timet

Преобразует значение datetime в метку времени Time_t (Unix).

```
DT2Timet([ tTime [, bLocalDateTime ]])
```

Параметры

tTime (необязательно)

Значение даты и времени.

bLocalDateTime (необязательно)

по умолчанию = .F.

Если переданное значение DateTime находится в локальном часовом поясе, передайте .T., если значение DateTime находится в UTC, передайте .F. или опустите параметр.

Возвращаемое значение

Метка времени Time_t (UNIX) — числовое значение, указывающее количество секунд, прошедших с 1 января 1970 года.

Замечания

Если вы не передадите функции никаких параметров, она вернет временную метку UNIX текущего времени.

Пример

```
? "Текущая временная метка UNIX", DT2Timet(), ;  
DT2Timet(DATETIME(), .T.)  
? "Текущая дата и время в местном часовом поясе", ;  
DATETIME(), TIMET2DT(DT2TIMET())  
? "Текущая дата и время в UTC", GetSystemTime (.T.), ;  
TIMET2DT(DT2TIMET(), .T.)
```

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

Используемые функции WinApi

[GetSystemTime](#)

[SystemTimeToFileTime](#)

DT2UTC

Преобразует значение даты и времени текущего активного часового пояса в значение даты и времени в формате UTC.

```
DT2UTC( tTime )
```

Параметры

tTime

Значение даты и времени.

Возвращаемое значение

Значение даты и времени в формате UTC.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

ExpandEnvironmentStringsEx

Расширяет переменные среды в переданной строке.

```
ExpandEnvironmentStringsEx( cEnvironmentString )
```

Параметры

cEnvironmentString

Строка, содержащая одну или несколько строк переменных среды в форме: %variableName%.

Для каждой такой ссылки часть %variableName% заменяется текущим значением этой переменной среды.

При поиске имени переменной среды регистр игнорируется. Если имя не найдено, часть %variableName% остается неразвернутой.

Возвращаемое значение

Расширенная строка среды.

Смотрите также

Ссылки

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[GetLocaleInfoEx](#)

[GetSystemDirectoryEx](#)

[GetWindowsDirectoryEx](#)

[OsEx](#)

Используемые функции WinApi

[ExpandEnvironmentStrings](#)

FChSizeEx

Изменяет размер файла, открытого с помощью функции [FOpenEx](#) или [FCreateEx](#).

```
FChSizeEx( nFileHandle , nNewFileSize )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

nNewFileSize

Новый размер файла.

Возвращаемое значение

Новый размер файла (числовой).

Смотрите также

Ссылки

[AFHandlesEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[SetEndOfFile](#)

[SetFilePointer](#)

FCloseEx

Сохраняет изменения в файле на диск и закрывает файл или порт связи, открытый с помощью функции [FCreateEx](#) или [FOpenEx](#).

```
FCloseEx( nFileHandle )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

Возвращаемое значение

.T. если файл был закрыт, .F. если произошла ошибка.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[CloseHandle](#)

FCreateEx

Создает и открывает файл.

```
FCreateEx( cFileName [, nAttributesAndFlags [, nAccessMode [, nShareMode ]]])
```

Параметры

cFileName

Указывает имя создаваемого файла. Вы можете включить обозначение диска и путь к имени файла. Если обозначение диска или путь не включены, файл создается в каталоге по умолчанию.

nAttributesAndFlags (необязательно, добавочный)

по умолчанию = FILE_ATTRIBUTE_NORMAL

Атрибуты и флаги файла или устройства, FILE_ATTRIBUTE_NORMAL является наиболее распространенным значением по умолчанию для файлов. Этот параметр может включать любую комбинацию доступных атрибутов файла (FILE_ATTRIBUTE_*). Все другие атрибуты файла переопределяют FILE_ATTRIBUTE_NORMAL. Этот параметр также может содержать комбинации флагов (FILE_FLAG_*) для управления поведением кэширования файла или устройства, режимами доступа и другими флагами специального назначения. Они сочетаются с любыми значениями FILE_ATTRIBUTE_*.

Атрибуты файла:

Значение	Описание
FILE_ATTRIBUTE_READONLY	Файл доступен только для чтения. Приложения могут читать файл, но не могут записывать в него или удалять его.
FILE_ATTRIBUTE_HIDDEN	Файл скрыт. Не включайте его в обычный список каталогов.
FILE_ATTRIBUTE_SYSTEM	Файл является частью операционной системы или используется исключительно ею.
FILE_ATTRIBUTE_ARCHIVE	Файл должен быть архивирован. Приложения используют этот атрибут для пометки файлов для резервного копирования или удаления.
FILE_ATTRIBUTE_NORMAL	Файл не имеет других установленных атрибутов. Этот атрибут действителен только при использовании в одиночку.
FILE_ATTRIBUTE_TEMPORARY	Файл используется для временного хранения.
FILE_ATTRIBUTE_OFFLINE	Данные файла не доступны немедленно. Этот атрибут указывает на то, что данные файла физически перемещены в автономное хранилище. Этот атрибут используется Remote Storage, иерархическим программным обеспечением для управления хранилищем. Приложения не должны произвольно изменять этот атрибут.
FILE_ATTRIBUTE_ENCRYPTED	Файл или каталог зашифрован. Для файла это означает, что все данные в файле зашифрованы. Для каталога это означает, что шифрование является значением по умолчанию для вновь созданных файлов и подкаталогов. Для получения дополнительной информации см. Шифрование файлов. Этот флаг не действует, если также указан FILE_ATTRIBUTE_SYSTEM.

Флаги:

Значение	Описание
FILE_FLAG_BACKUP_SEMANTICS	<p>Файл открывается или создается для операции резервного копирования или восстановления. Система гарантирует, что вызывающий процесс переопределяет проверки безопасности файла, если у процесса есть привилегии SE_BACKUP_NAME и SE_RESTORE_NAME.</p> <p>Вы должны установить этот флаг, чтобы получить дескриптор каталога. Дескриптор каталога может быть передан некоторым функциям вместо дескриптора файла. Для получения дополнительной информации см. раздел Примечания.</p>
FILE_FLAG_DELETE_ON_CLOSE	<p>Файл должен быть удален немедленно после закрытия всех его дескрипторов, включая указанный дескриптор и любые другие открытые или дублированные дескрипторы.</p> <p>Если существуют открытые дескрипторы файла, вызов завершается неудачей, если только все они не были открыты с режимом общего доступа FILE_SHARE_DELETE. Последующие запросы на открытие файла завершаются неудачей, если только не указан режим общего доступа FILE_SHARE_DELETE.</p>
FILE_FLAG_NO_BUFFERING	<p>Файл или устройство открывается без системного кэширования для чтения и записи данных. Этот флаг не влияет на кэширование жесткого диска или файлы, отображенные в памяти. Для успешной работы с файлами, открытыми с помощью CreateFile с использованием флага FILE_FLAG_NO_BUFFERING, существуют строгие требования.</p>
FILE_FLAG_OPEN_NO_RECALL	<p>Данные файла запрошены, но они должны продолжать находиться в удаленном хранилище. Они не должны транспортироваться обратно в локальное хранилище. Этот флаг предназначен для использования удаленными системами хранения.</p>
FILE_FLAG_POSIX_SEMANTICS	<p>Доступ будет осуществляться в соответствии с правилами POSIX. Это включает разрешение нескольких файлов с именами, отличающимися только регистром, для файловых систем, которые поддерживают такое именование. Будьте осторожны при использовании этой опции, поскольку файлы, созданные с этим флагом, могут быть недоступны для приложений, написанных для MS-DOS или 16-разрядной Windows.</p>
FILE_FLAG_RANDOM_ACCESS	<p>Доступ предполагается случайным. Система может использовать это как подсказку для оптимизации кэширования файлов.</p> <p>Этот флаг не имеет эффекта, если файловая система не поддерживает кэшированный ввод-вывод и FILE_FLAG_NO_BUFFERING.</p>
FILE_FLAG_SEQUENTIAL_SCAN	<p>Доступ должен быть последовательным от начала до конца. Система может использовать это как подсказку для оптимизации кэширования файлов.</p> <p>Этот флаг не следует использовать, если будет использоваться чтение с задержкой (то есть обратное сканирование). Этот флаг не имеет эффекта, если файловая система не поддерживает кэшированный</p>

	ввод-вывод и FILE_FLAG_NO_BUFFERING.
FILE_FLAG_WRITE_THROUGH	Операции записи не будут проходить через промежуточный кэш, они будут отправляться непосредственно на диск.

Примечание

Более подробную информацию об атрибутах и флагах можно найти в документации [CreateFile](#).

nAccessMode (необязательно)

по умолчанию = 0

допустимые значения:

0 - только чтение

1 - только запись

2 - чтение/запись

nSharemode (необязательно)

по умолчанию = 0 (без обмена)

Одно или комбинация следующих значений.

Sharemode	Описание
0	Запрещает другим процессам открывать файл или устройство, если они запрашивают доступ на удаление, чтение или запись.
FILE_SHARE_READ	Позволяет последующим операциям открытия файла или устройства запрашивать доступ для чтения. В противном случае другие процессы не смогут открыть файл или устройство, если они запросят доступ для чтения. Если этот флаг не указан, но файл или устройство были открыты для доступа для чтения, функция завершается ошибкой.
FILE_SHARE_WRITE	Позволяет последующим операциям открытия файла или устройства запрашивать доступ на запись. В противном случае другие процессы не смогут открыть файл или устройство, если они запросят доступ на запись. Если этот флаг не указан, но файл или устройство были открыты для доступа на запись или имеют сопоставление файлов с доступом на запись, функция завершается ошибкой.
FILE_SHARE_DELETE	Позволяет последующим операциям открытия файла или устройства запрашивать доступ для удаления. В противном случае другие процессы не смогут открыть файл или устройство, если они запросят доступ для удаления. Если этот флаг не указан, но файл или устройство были открыты для доступа для удаления, функция завершается ошибкой. Примечание: доступ для удаления позволяет как операции удаления, так и операции переименования.

Возвращаемое значение

Значение HANDLE (числовое) для созданного файла или -1, если произошла ошибка.

Замечания

Параметры *nAccessMode* и *nShareMode* расширили функциональность собственного [FCREATE](#)() FoxPro.

Если файл с указанным вами именем уже существует, он перезаписывается без предупреждения.

FCreateEx() назначает файлу номер дескриптора файла, который можно использовать для идентификации файла в других низкоуровневых функциях файлов Visual FoxPro. FCreateEx() возвращает номер дескриптора файла при создании файла или возвращает -1, если файл не может быть создан.

Пример

Создайте INI-файл, доступный только для чтения, в текущем каталоге, если он еще не существует:

```
LOCAL cIniFile, nFileHandle
cIniFile = [./MyFile.ini]
IF NOT FILE(cIniFile)  && Существует ли файл?
    nFileHandle = FCreateEx(cIniFile)  && Если нет, создайте
    его в режиме только для чтения
    IF nFileHandle < 0 && Проверка на наличие ошибок при
    открытии файла
        WAIT "Невозможно открыть или создать файл INI" WINDOW
        NOWAIT
    ELSE  && Если нет ошибок, записать в файл
        FWriteEx(nFileHandle, "[Program Information]" + ;
        CHR(13) + "DataDir="+ADDBS(GETENV("APPDATA")) + "MyApp\")
    ENDIF
    FCloseEx(nFileHandle)  && Закрыть файл
ENDIF
MODIFY FILE (cIniFile) NOWAIT  && Открыть файл в окне
редактирования
```

Смотрите также

Ссылки

[AFHandlesEx](#)
[FChSizeEx](#)
[FCloseEx](#)
[FEoFEx](#)
[FFlushEx](#)
[FGetsEx](#)
[FLockFile](#)
[FLockFileEx](#)
[FOpenEx](#)
[FPutsEx](#)
[FReadEx](#)
[FSeekEx](#)
[FUnlockFile](#)
[FUnlockFileEx](#)
[FWriteEx](#)

Используемые функции WinApi

[CreateFile](#)

FEoFEx

Определяет, находится ли указатель файла в конце файла.

```
FEoFEx( nFileHandle )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

Возвращаемое значение

.T. если указатель файла находится в EOF, .F. в противном случае.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[SetFilePointer](#)

FFlushEx

Сбрасывает на диск файл, открытый с помощью [FCreateEx](#) или [FOpenEx](#).

```
FFlushEx( nFileHandle )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

Возвращаемое значение

.Т. если вызов прошел успешно, .F. в противном случае.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEoFEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[FlushFileBuffers](#)

FGetsEx

Возвращает последовательность байтов из указанного файла или порта связи, открытого с помощью [FOpenEx](#) или [FCreateEx](#), до тех пор, пока не встретится возврат каретки.

```
FGetsEx( nFileHandle [, nMaxBytesToRead ] )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

nMaxBytesToRead (необязательно)

по умолчанию = 256

Максимальное количество байтов для чтения.

Возвращаемое значение

Считанные данные.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEOFEx](#)

[FFlushEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[ReadFile](#)

[SetFilePointer](#)

FindFileChange

Отслеживает изменения файлов в каталоге в отдельном потоке.

```
FindFileChange( cDirectory , bWatchSubtree , nFilter , cCallback )
```

Параметры

cDirectory

Полный путь к отслеживаемому каталогу. Это не может быть относительный путь или пустая строка.

Имя ограничено MAX_PATH (260) символами.

bWatchSubtree

Если этот параметр равен .T., функция отслеживает дерево каталогов, корнем которого является указанный каталог; если он равен .F., функция отслеживает только указанный каталог.

nFilter

Условия фильтра, которые удовлетворяют уведомлению об изменении. Этот параметр может иметь одно или несколько из следующих значений.

Значение	Значение
FILE_NOTIFY_CHANGE_FILE_NAME	Любое изменение имени файла в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении. Изменения включают переименование, создание или удаление имени файла.
FILE_NOTIFY_CHANGE_DIR_NAME	Любое изменение имени каталога в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении. Изменения включают создание или удаление каталога.
FILE_NOTIFY_CHANGE_ATTRIBUTES	Любое изменение атрибута в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении.
FILE_NOTIFY_CHANGE_SIZE	Любое изменение размера файла в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении. Операционная система обнаруживает изменение размера файла только тогда, когда файл записывается на диск. Для операционных систем, использующих обширное кэширование, обнаружение происходит только тогда, когда кэш достаточно очищен.
FILE_NOTIFY_CHANGE_LAST_WRITE	Любое изменение последнего времени записи файлов в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении. Операционная система обнаруживает изменение последнего времени записи только тогда, когда файл записывается на диск. Для операционных систем, использующих обширное кэширование, обнаружение происходит только тогда, когда кэш достаточно очищен.
FILE_NOTIFY_CHANGE_SECURITY	Любое изменение дескриптора безопасности в отслеживаемом каталоге или поддереве вызывает оператор ожидания уведомления об изменении

cCallback

Функция, которая вызывается, когда в контролируемом каталоге или поддереве возникает одно из условий фильтра.

Функция должна иметь следующую сигнатуру:

```
FUNCTION FolderChanged
    LPARAMETERS hHandle, cPath, nError
    IF nError = 0
        ? cPath + ' изменен'
    ELSE
        ? 'Функция: ' + cPath + " не удалась. ErrorNo. ", ;
        nError
    ENDIF
ENDFUNC
```

Возвращаемое значение

Числовой дескриптор, представляющий поток, который отслеживает каталог.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[FindFirstChangeNotification](#)
[FindNextChangeNotification](#)
[CreateThread](#)
[WaitForMultipleObjects](#)
[PostMessage](#)

FindFileChangeEx

Отслеживает изменения файлов в каталоге в отдельном потоке.

```
FindFileChangeEx( cPath , bWatchSubtree , nFilter , cCallback [,
oCallbackObject ] )
```

Параметры

cDirectory

Полный путь к отслеживаемому каталогу. Это не может быть относительный путь или пустая строка.
Имя ограничено MAX_PATH (260) символами.

bWatchSubtree

Если этот параметр равен .T., функция отслеживает дерево каталогов, корнем которого является указанный каталог; если он равен .F., функция отслеживает только указанный каталог.

nFilter

Условия фильтра, которые удовлетворяют уведомлению об изменении. Этот параметр может иметь одно или несколько из следующих значений.

Значение	Значение
FILE_NOTIFY_CHANGE_FILE_NAME	Любое изменение имени файла в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении. Изменения включают переименование, создание или удаление имени файла.
FILE_NOTIFY_CHANGE_DIR_NAME	Любое изменение имени каталога в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении. Изменения включают создание или удаление каталога.
FILE_NOTIFY_CHANGE_ATTRIBUTES	Любое изменение атрибута в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении.
FILE_NOTIFY_CHANGE_SIZE	Любое изменение размера файла в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении. Операционная система обнаруживает изменение размера файла только тогда, когда файл записывается на диск. Для операционных систем, использующих обширное кэширование, обнаружение происходит только тогда, когда кэш достаточно очищен.
FILE_NOTIFY_CHANGE_LAST_WRITE	Любое изменение последнего времени записи файлов в отслеживаемом каталоге или поддереве приводит к возврату операции ожидания уведомления об изменении. Операционная система обнаруживает изменение последнего времени записи только тогда, когда файл записывается на диск. Для операционных систем, использующих обширное кэширование, обнаружение происходит только тогда, когда кэш достаточно очищен.
FILE_NOTIFY_CHANGE_SECURITY	Любое изменение дескриптора безопасности в отслеживаемом каталоге или поддереве вызывает оператор ожидания уведомления об изменении

cCallback

Функция, которая вызывается, когда в контролируемом каталоге или поддереве возникает одно из условий фильтра.

Функция должна иметь следующую сигнатуру:

```
FUNCTION FolderChanged
    LPARAMETERS hHandle, nReason, cPath, cPath2
    DO CASE
        CASE m.nReason = 1
            ? 'Файл добавлен: ' + m.cPath
        CASE m.nReason = 2
            ? 'Файл удален: ' + m.cPath
        CASE m.nReason = 3
            ? 'Файл изменен: ' + m.cPath
        CASE m.nReason = 4
            ? 'Файл переименован из: ' + m.cPath + CHR(13) + ;
              CHR(10) + ' в: ' + m.cPath2
        CASE m.nReason = 0
            ? 'Ошибка просмотра пути: ' + m.cPath + ;
              " - ErrorNo: " + m.cPath2
            && ошибка - либо GetLastError из ReadDirectoryChangesW,
            && либо E_INSUFEMORY в ситуации нехватки памяти
    ENDCASE
ENDFUNC
```

oCallbackObject (необязательно)

ссылка на объект, на котором вызывается указанная функция обратного вызова.

Если объект не передан, функция обратного вызова должна быть глобальной функцией, если передан, то это метод на переданном объекте.

Возвращаемое значение

Числовой дескриптор, который идентифицирует каталог watches.

Передайте этот дескриптор в CancelFindFileChangeEx, чтобы остановить мониторинг каталога.

Смотрите также

Используемые функции WinApi

[ReadDirectoryChangesW](#)

[CreateIoCompletionPort](#)

[GetQueuedCompletionStatus](#)

[PostQueuedCompletionStatus](#)

[CancelIo](#)

[CreateThread](#)

[PostMessage](#)

FindRegistryChange

Отслеживает изменения ключа реестра в отдельном потоке.

```
FindRegistryChange( nRegKey , cKeyName , bWatchSubtree , nFilter ,
cCallback )
```

Параметры

nRegKey

Одна из следующих ключевых констант.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и устанавливается при первой ссылке процесса на</p>

	<p>HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>
HKEY_PERFORMANCE_NLSTEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe.</p> <p>Windows 2000: этот ключ не поддерживается.</p>
HKEY_PERFORMANCE_TEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на американском английском. Эти записи недоступны для</p>

	Regedit.exe и Regedt32.exe.
	Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

cKeyName

Имя раздела реестра, который необходимо отслеживать.

Примечание

Названия ключей не чувствительны к регистру.

bWatchSubtree

Если этот параметр равен .Т., функция сообщает об изменениях в указанном ключе и его подключах. Если параметр равен .F., функция сообщает об изменениях только в указанном ключе.

nFilter

Значение, указывающее изменения, о которых следует сообщить.

Этот параметр может быть одним или комбинацией следующих значений.

Значение	Значение
REG_NOTIFY_CHANGE_NAME	Уведомить вызывающего абонента о добавлении или удалении подключаемого ключа.
REG_NOTIFY_CHANGE_ATTRIBUTES	Уведомить вызывающего абонента об изменениях атрибутов ключа, таких как информация о дескрипторе безопасности.
REG_NOTIFY_CHANGE_LAST_SET	Уведомить вызывающего абонента об изменениях значения ключа. Это может включать добавление или удаление значения или изменение существующего значения.
REG_NOTIFY_CHANGE_SECURITY	Уведомить вызывающего абонента об изменениях в дескрипторе безопасности ключа.

cCallback

Функция, которая вызывается, когда одно из условий фильтра происходит в контролируемом разделе реестра или поддереве.

Функция должна иметь следующую сигнатуру:

```

FUNCTION RegistryKeyChanged
    LPARAMETERS hHandle, nError
    IF nError = 0
        ? 'RegistryKey изменен'
    ELSE
        ? 'Функция API не выполнена - ErrorNo. ', nError
    ENDIF
ENDFUNC

```

Возвращаемое значение

Числовой дескриптор, представляющий поток, который отслеживает раздел реестра.

Смотрите также

Ссылки

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegNotifyChangeKeyValue](#)
[RegOpenKeyEx](#)
[CreateThread](#)
[WaitForMultipleObjects](#)
[PostMessage](#)

Float2Str

Преобразует значение с плавающей точкой (32-битное числовое значение) в двоичную строку.

```
Float2Str( nValue )
```

Параметры

nValue

Числовое значение в диапазоне 3,4E +/- 38 (7 цифр).

Возвращаемое значение

Строка, которая в двоичном виде равна переданному значению с плавающей точкой.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

FLockFile

Блокирует область байтов в файле, открытом с помощью [FCreateEx](#) или [FOpenEx](#).

```
FLockFile( nFileHandle , nLockOffset , nBytesToLock )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

nLockOffset

Начальное смещение раздела байтов для блокировки в файле.

nBytesToLock

Количество байтов, которые необходимо заблокировать в файле.

Возвращаемое значение

.T. если блокировка прошла успешно, .F. в противном случае.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEOFEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[LockFile](#)

FLockFileEx

Блокирует область байтов в файле, открытом с помощью [FCreateEx](#) или [FOpenEx](#).

```
FLockFileEx( nFileHandle , nLockOffset , nBytesToLock [,  
nLockMode ])
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

nLockOffset

Начальное смещение раздела байтов для блокировки в файле.

nBytesToLock

Количество байтов, которые необходимо заблокировать в файле.

nLockMode (необязательно, аддитивно)

по умолчанию = 0

Режим блокировки	Описание
LOCKFILE_FAIL_IMMEDIATELY	Функция немедленно возвращается, если не может получить запрошенную блокировку. Если этот флаг не указан, функция ждет.
LOCKFILE_EXCLUSIVE_LOCK	Функция запрашивает исключительную блокировку. Если этот флаг не указан, функция запрашивает общую блокировку.

Возвращаемое значение

.T. если блокировка прошла успешно, .F. в противном случае.

Смотрите также

Ссылки

[AFHandlesEx](#)
[FChSizeEx](#)
[FCloseEx](#)
[FCreateEx](#)
[FEofEx](#)
[FFlushEx](#)
[FGetsEx](#)
[FLockFile](#)
[FOpenEx](#)
[FPutsEx](#)
[FReadEx](#)
[FSeekEx](#)
[FUnlockFile](#)
[FUnlockFileEx](#)
[FWriteEx](#)

Используемые функции WinApi

[LockFileEx](#)

FOpenEx

Открывает файл.

```
FOpenEx( cFileName [, nAttributesAndFlags [, nAccessMode [,
nShareMode ]]])
```

Параметры

cFileName

Имя файла, который вы хотите открыть.

Может быть задан относительным, абсолютным или с помощью UNC.

nAttributesAndFlags (необязательно, добавочный)

Указывает, как открыть файл.

Значение	Описание
FILE_FLAG_BACKUP_SEMANTICS	Файл открывается или создается для операции резервного копирования или восстановления. Система гарантирует, что вызывающий процесс переопределяет проверки безопасности файла, если у процесса есть привилегии SE_BACKUP_NAME и SE_RESTORE_NAME. Вы должны установить этот флаг, чтобы получить дескриптор каталога. Дескриптор каталога может быть передан некоторым функциям вместо дескриптора файла. Для получения дополнительной информации см. раздел Примечания.
FILE_FLAG_DELETE_ON_CLOSE	Файл должен быть удален немедленно после закрытия всех его дескрипторов, включая указанный дескриптор и любые другие открытые или дублированные дескрипторы. Если существуют открытые дескрипторы файла, вызов завершается неудачей, если только все они не были открыты с режимом общего доступа FILE_SHARE_DELETE. Последующие запросы на открытие файла завершаются неудачей, если только не указан режим общего доступа FILE_SHARE_DELETE.
FILE_FLAG_NO_BUFFERING	Файл или устройство открывается без системного кэширования для чтения и записи данных. Этот флаг не влияет на кэширование жесткого диска или файлы, отображенные в памяти. Для успешной работы с файлами, открытыми с помощью CreateFile с использованием флага FILE_FLAG_NO_BUFFERING, существуют строгие требования.
FILE_FLAG_OPEN_NO_RECALL	Данные файла запрошены, но они должны продолжать находиться в удаленном хранилище. Они не должны транспортироваться обратно в локальное хранилище. Этот флаг предназначен для использования удаленными системами хранения.
FILE_FLAG_POSIX_SEMANTICS	Доступ будет осуществляться в соответствии с правилами POSIX. Это включает разрешение нескольких файлов с именами, отличающимися только регистром, для файловых систем, которые поддерживают такое именование. Будьте осторожны при использовании этой опции, поскольку файлы, созданные с этим флагом, могут быть недоступны для приложений, написанных для MS-DOS или 16-разрядной Windows.

FILE_FLAG_RANDOM_ACCESS	Доступ предполагается случайным. Система может использовать это как подсказку для оптимизации кэширования файлов. Этот флаг не имеет эффекта, если файловая система не поддерживает кэшированный ввод-вывод и FILE_FLAG_NO_BUFFERING.
FILE_FLAG_SEQUENTIAL_SCAN	Доступ должен быть последовательным от начала до конца. Система может использовать это как подсказку для оптимизации кэширования файлов. Этот флаг не следует использовать, если будет использоваться чтение с задержкой (то есть обратное сканирование). Этот флаг не имеет эффекта, если файловая система не поддерживает кэшированный ввод-вывод и FILE_FLAG_NO_BUFFERING.
FILE_FLAG_WRITE_THROUGH	Операции записи не будут проходить через промежуточный кэш, они будут отправляться непосредственно на диск.

Примечание

Более подробную информацию о флагах можно найти в документации [CreateFile](#).

nAccessMode (необязательно)

по умолчанию = 0

допустимые значения:

0 - только чтение

1 - только запись

2 - чтение/запись

nSharemode (необязательно)

по умолчанию = 0 (без обмена)

Одно или комбинация следующих значений.

Sharemode	Описание
0	Запрещает другим процессам открывать файл или устройство, если они запрашивают доступ на удаление, чтение или запись.
FILE_SHARE_READ	Позволяет последующим операциям открытия файла или устройства запрашивать доступ для чтения. В противном случае другие процессы не смогут открыть файл или устройство, если они запросят доступ для чтения. Если этот флаг не указан, но файл или устройство были открыты для доступа для чтения, функция завершается ошибкой.
FILE_SHARE_WRITE	Позволяет последующим операциям открытия файла или устройства запрашивать доступ на запись. В противном случае другие процессы не смогут открыть файл или устройство, если они запросят доступ на запись. Если этот флаг не указан, но файл или устройство были открыты для доступа на запись или имеют сопоставление файлов с доступом на запись, функция завершается ошибкой.
FILE_SHARE_DELETE	Позволяет последующим операциям открытия файла или устройства запрашивать доступ для удаления. В противном случае другие процессы не смогут открыть файл или устройство, если они запросят доступ для

удаления. Если этот флаг не указан, но файл или устройство были открыты для доступа для удаления, функция завершается ошибкой. Примечание: доступ для удаления позволяет как операции удаления, так и операции переименования.

Возвращаемое значение

Возвращает дескриптор (целое число) файла. Положительное целое число указывает на то, что файл был открыт успешно, а -1 указывает на неудачу.

Пример

```
LOCAL nFileHandle
nFileHandle=FOPENEX("c:\temp\audit.log")
? nFileHandle
&& Теперь вы можете управлять этим файлом, используя
&& nFileHandle в качестве параметра в других функциях.
```

Смотрите также

Ссылки

[AFHandlesEx](#)
[FChSizeEx](#)
[FCloseEx](#)
[FCreateEx](#)
[FEofEx](#)
[FFlushEx](#)
[FGetsEx](#)
[FLockFile](#)
[FLockFileEx](#)
[FPutsEx](#)
[FReadEx](#)
[FSeekEx](#)
[FUnlockFile](#)
[FUnlockFileEx](#)
[FWriteEx](#)

Используемые функции WinApi

[CreateFile](#)

FormatMessageEx

Извлекает сообщение об ошибке для указанного номера ошибки.

```
FormatMessageEx( nLastError [, nLanguageId [, nModuleHandle ]])
```

Параметры

nLastError

Номер ошибки.

nLanguageId (необязательно)

по умолчанию = 0

Идентификатор языка для запрошенного сообщения.

Если вы передадите в этом параметре определенный LANGID, FormatMessageEx вернет сообщение только для этого LANGID. Если функция не может найти сообщение для этого LANGID, она устанавливает Last-Error в ERROR_RESOURCE_LANG_NOT_FOUND. Если вы передадите ноль, FormatMessageEx ищет сообщение для LANGID в следующем порядке:

Нейтральный к языку LANGID потока, на основе значения локали потока LANGID пользователя по умолчанию, на основе значения локали пользователя по умолчанию LANGID системы по умолчанию, на основе значения локали системы по умолчанию US English

nModuleHandle (необязательно)

по умолчанию = 0

Дескриптор модуля, содержащего таблицу сообщений для поиска.

Возвращаемое значение

Сообщение об ошибке.

Смотрите также

Ссылки

[AErrorEx](#)
[VFP2CSys](#)

Используемые функции WinApi

[FormatMessage](#)

FputsEx

Записывает строку символов, возврат каретки и перевод строки в файл, открытый с помощью [FCreateEx](#) или [FOpenEx](#).

```
FputsEx( nFileHandle , cData [, nMaxBytesToWrite ] )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

cData

Содержимое для записи в файл.

nMaxBytesToWrite (необязательно)

по умолчанию = [LEN](#)(*cData*)

Максимальное количество байтов для записи.

Возвращаемое значение

Количество записанных байтов или 0, если произошла ошибка.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEofEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[WriteFile](#)

FReadEx

Возвращает указанное количество байтов из файла, открытого с помощью [FCreateEx](#) или [FOpenEx](#).

```
FReadEx( nFileHandle , nBytesToRead )
```

Параметры

nFileHandle

Указывает номер дескриптора файла, из которого FReadEx возвращает данные. Вы можете получить *nFileHandle* из [FOpenEx](#), [FCreateEx](#) или функции Windows API.

nBytesToRead

Указывает количество возвращаемых байтов. FReadEx возвращает данные, начиная с текущей позиции указателя файла, и продолжает, пока не вернет *nBytesToRead* байтов или пока не встретит конец файла.

Возвращаемое значение

Данные, считанные из файла, или пустая строка, если произошла ошибка.

Замечания

НЕЛЬЗЯ смешивать и сопоставлять дескриптор файла собственных функций VFP с дескриптором файла функций [VFP2C32](#), поскольку дескриптор файла, созданный с помощью одного из них, не будет работать с другим.

Пример

Откройте файл и верните первые 20 байт:

```
LOCAL nFileHandle, cBytesRead
nFileHandle = FOpenEx("c:\temp\audit.log")
cBytesRead = FReadEx(nFileHandle, 20)
? cBytesRead && Возвращает первые 20 байт файла, так как
  && позиция чтения не была перемещена после FOpenEx()
```

Смотрите также

Ссылки

[AFHandlesEx](#)
[FChSizeEx](#)
[FCloseEx](#)
[FCreateEx](#)
[FEofEx](#)
[FFlushEx](#)
[FGetsEx](#)
[FLockFile](#)
[FLockFileEx](#)
[FOpenEx](#)
[FPutsEx](#)
[FSeekEx](#)
[FUnlockFile](#)
[FUnlockFileEx](#)
[FWriteEx](#)

Используемые функции WinApi

[ReadFile](#)

FreeHGlobal

Освобождает память, выделенную с помощью [AllocHGlobal](#).

```
FreeHGlobal( nHandle )
```

Параметры

nHandle

Дескриптор или указатель на блок памяти, возвращаемый [AllocHGlobal](#).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[GlobalFree](#)

FreeMem

Освобождает блок памяти, выделенный из внутренней кучи библиотеки функцией [AllocMem](#) или [ReAllocMem](#).

```
FreeMem( nAddress )
```

Параметры

nAddress

Указатель на блок памяти, выделенный [AllocMem](#).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapFree](#)

FreePMem

Освобождает блок памяти, на который указывает переданный указатель, из внутренней кучи библиотеки.

```
FreePMem( nAddress )
```

Параметры

nAddress

Указатель на ячейку памяти, указатель в этой ячейке освобождается.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapFree](#)

FreeRefArray

Освобождает всю память, выделенную для переданного массива в стиле С.

```
FreeRefArray( nAddress , nStartElement , nElements )
```

Параметры

Возвращаемое значение

.Т. если все элементы были успешно освобождены, .F. в противном случае.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapFree](#)

FSeekEx

Перемещает указатель файла в файле, открытом с помощью [FCreateEx](#) или [FOpenEx](#).

```
FSeekEx( nFileHandle , nBytesToMove [, nRelativePosition ])
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

nBytesToMove

Количество байтов, на которое следует переместить указатель файла.

nRelativePosition (необязательно)

по умолчанию = FILE_BEGIN

Одно из следующих значений.

Значение	Описание
FILE_BEGIN	Начальной точкой является ноль или начало файла.
FILE_CURRENT	Начальной точкой является текущее значение указателя файла.
FILE_END	Начальной точкой является текущая позиция конца файла.

Возвращаемое значение

Новое положение указателя файла.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEofEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[SetFilePointer](#)

FT2DT

Преобразует структуру [FILETIME](#) в значение datetime.

```
FT2DT( nFileTimePointer [, bToLocalTime ] )
```

Параметры

nFileTimePointer

Указатель на структуру [FILETIME](#).

bToLocalTime (необязательно)

по умолчанию = .F.

Если .T., то время файла преобразуется в местный часовой пояс.

Возвращаемое значение

Значение даты и времени.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FI](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

FUnlockFile

Разблокирует область байтов в файле, открытом с помощью [FCreateEx](#) или [FOpenEx](#).

```
FUnlockFile( nFileHandle , nLockOffset , nBytesToLock )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

nLockOffset

Начальное смещение раздела байтов для блокировки в файле.

nBytesToLock

Количество байтов, которые необходимо заблокировать в файле.

Возвращаемое значение

.T. если байтовый диапазон был успешно разблокирован, .F. в противном случае.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEofEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFileEx](#)

[FWriteEx](#)

Используемые функции WinApi

[UnlockFile](#)

FUnlockFileEx

Разблокирует область байтов в файле, открытом с помощью [FCreateEx](#) или [FOpenEx](#).

```
FUnlockFileEx( nFileHandle , nLockOffset , nBytesToLock )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

nLockOffset

Начальное смещение раздела байтов для блокировки в файле.

nBytesToLock

Количество байтов, которые необходимо заблокировать в файле.

Возвращаемое значение

.T. если байтовый диапазон был успешно разблокирован, .F. в противном случае.

Смотрите также

Ссылки

[AFHandlesEx](#)
[FChSizeEx](#)
[FCloseEx](#)
[FCreateEx](#)
[FEofEx](#)
[FFlushEx](#)
[FGetsEx](#)
[FLockFile](#)
[FLockFileEx](#)
[FOpenEx](#)
[FPutsEx](#)
[FReadEx](#)
[FSeekEx](#)
[FUnlockFile](#)
[FWriteEx](#)

Используемые функции WinApi

[UnlockFileEx](#)

FWriteEx

Записывает строку символов в файл, открытый с помощью [FCreateEx](#) или [FOpenEx](#).

```
FWriteEx( nFileHandle , cData [, nMaxBytesToWrite ] )
```

Параметры

nFileHandle

Дескриптор файла, полученный из [FCreateEx](#), [FOpenEx](#) или функции Windows API.

cData

Содержимое для записи в файл.

nMaxBytesToWrite

по умолчанию = [LEN](#)(cData)

Максимальное количество байтов для записи в файл.

Возвращаемое значение

Количество записанных байтов или 0, если произошла ошибка.

Смотрите также

Ссылки

[AFHandlesEx](#)

[FChSizeEx](#)

[FCloseEx](#)

[FCreateEx](#)

[FEofEx](#)

[FFlushEx](#)

[FGetsEx](#)

[FLockFile](#)

[FLockFileEx](#)

[FOpenEx](#)

[FPutsEx](#)

[FReadEx](#)

[FSeekEx](#)

[FUnlockFile](#)

[FUnlockFileEx](#)

Используемые функции WinApi

[WriteFile](#)

GetCursorPosEx

Извлекает положение курсора мыши.

```
GetCursorPosEx( @nXCoord , @nYCoord [, bRelative [, nHwnd |  
cWindowName ]])
```

Параметры

@nXCoord

Переменная по ссылке, в которой хранится координата X курсора.

@nYCoord

Переменная по ссылке, в которой хранится координата Y курсора.

bRelative (необязательно)

Если опустить bRelative или передать .F., координаты будут вычисляться относительно всего экрана,
в противном случае координаты будут вычисляться относительно указанного окна,
переданного в параметре nHwnd | cWindowName .

nHwnd | cWindowName (необязательно)

Либо hWnd, либо имя окна VFP. Координаты вычисляются относительно этого окна.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AFontInfo](#)
[ASplitStr](#)
[Decimals](#)
[Int64_Add](#)
[Int64_Div](#)
[Int64_Mod](#)
[Int64_Mul](#)
[Int64_Sub](#)

Используемые функции WinApi

[GetCursorPos](#)
[ScreenToClient](#)

GetFileAttributesEx2

Извлекает атрибуты файла или каталога.

```
GetFileAttributesEx2( cFileName [, bStringAttributes ] )
```

Параметры

cFileName

Имя файла, для которого необходимо получить атрибуты.

Примечание

Это может быть относительный путь или только имя файла, функция VFP [FULLPATH](#) используется для получения полного имени файла.

Используется обычный путь поиска VFP ([SET PATH](#)).

bStringAttributes (необязательно)

по умолчанию = .F.

Если .T. атрибуты файла возвращаются в строковой форме.

Возвращаемое значение

Числовое значение, представляющее атрибуты файла.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[GetFileAttributes](#)

GetFileOwner

Возвращает владельца файла.

```
GetFileOwner( cFileName , @rOwner [, @rDomain [, @rSidType ]])
```

Параметры

cFileName

Имя файла, для которого необходимо получить владельца.

Примечание

Это может быть относительный путь или только имя файла, функция VFP [FULLPATH](#) используется для получения полного имени файла.
Используется обычный путь поиска VFP ([SET PATH](#)).

rOwner

Ссылка на переменную, которая получает владельца файла.

rDomain

Ссылка на переменную, которая получает домен владельца файла.

rSidType

Ссылка на переменную, которая получает тип учетной записи.
Тип — одно из значений в перечислении [SID_NAME_USE](#) .

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[CreateFile](#)
[GetKernelObjectSecurity](#)
[GetSecurityDescriptorOwner](#)
[LookupAccountSid](#)
[CloseHandle](#)

GetFileSizeEx2

Возвращает размер файла.

```
GetFileSizeEx2 ( cFileName )
```

Параметры

cFileName

Имя файла, для которого необходимо получить размер.

Примечание

Это может быть относительный путь или только имя файла, функция VFP [FULLPATH](#) используется для получения полного имени файла.
Используется обычный путь поиска VFP ([SET PATH](#)).

Возвращаемое значение

Размер файла.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[CreateFile](#)
[GetFileSizeEx](#) или
[GetFileSize](#)
[CloseHandle](#)

GetFileTimes

Возвращает время создания, последнего доступа и последней записи файла.

```
GetFileTimes( cFileName , @rCreationTime [, @rLastAccessTime [,  
@rLastWriteTime [, bUTCTimes ]]])
```

Параметры

cFileName

Имя файла или каталога, для которого требуется получить время файла.

Примечание

Это может быть относительный путь или только имя файла, функция VFP [FULLPATH](#) используется для получения полного имени файла.

Используется обычный путь поиска VFP ([SET PATH](#)).

rCreationTime

Переменная по ссылке, в которой хранится время создания файла.

Примечание

Вы также можете передать NULL для любого параметра времени, если его не нужно извлекать.

rLastAccessTime (необязательно)

Переменная по ссылке, в которой хранится время последнего доступа к файлу.

rLastWriteTime (необязательно)

Переменная по ссылке, в которой хранится время последней записи файла.

bUTCTimes (необязательно)

по умолчанию = .F.

Если .T., то время файла возвращается в формате UTC, если .F., то оно преобразуется в местный часовой пояс.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)

[SetFileAttributesEx](#)[SetFileTimes](#)**Используемые функции WinApi**[GetFileTime](#)[CreateFile](#)[CloseHandle](#)

GetIUnknown

Возвращает указатель на интерфейс IUnknown объекта COM.

```
GetIUnknown( cObjectName )
```

Параметры

cObjectName

Имя переменной, ссылающейся на COM-объект в виде строки, для которой необходимо получить указатель IUnknown.

Возвращаемое значение

Указатель (числовой) на интерфейс IUnknown переданного COM-объекта.

Смотрите также

Ссылки

[CLSIDFromProgID](#)

[CLSIDFromString](#)

[CreateGuid](#)

[CreateThreadObject](#)

[IsEqualGuid](#)

[ProgIDFromCLSID](#)

[RegisterActiveObject](#)

[RegisterObjectAsFileMoniker](#)

[RevokeActiveObject](#)

[StringFromCLSID](#)

GetLocaleInfoEx

Извлекает информацию о локали.

```
GetLocaleInfoEx( nType [, nLocaleID ] )
```

Параметры

nType

nType может принимать значения от 1 до 95.

nLocaleID (необязательно)

по умолчанию = LOCALE_USER_DEFAULT

Идентификатор локали, для которого необходимо получить информацию. Вы можете использовать макрос MAKELCID для создания идентификатора локали или использовать одно из следующих предопределенных значений.

LOCALE_CUSTOM_DEFAULT
LOCALE_CUSTOM_UI_DEFAULT
LOCALE_CUSTOM_UNSPECIFIED
LOCALE_INVARIANT
LOCALE_SYSTEM_DEFAULT
LOCALE_USER_DEFAULT

Возвращаемое значение

Информация (строка), полученная из API [GetLocaleInfo](#).

Смотрите также

Ссылки

[ADesktopArea](#)
[ADesktops](#)
[ADisplayDevices](#)
[AResolutions](#)
[AWindowStations](#)
[ExpandEnvironmentStringsEx](#)
[GetSystemDirectoryEx](#)
[GetWindowsDirectoryEx](#)
[OsEx](#)

Используемые функции WinApi

[GetLocaleInfo](#)

GetLongPathNameEx

Преобразует указанный путь в его длинную форму.

```
GetLongPathNameEx( cFileName )
```

Параметры

cFileName

Путь, который необходимо преобразовать.

Примечание

Имя ограничено MAX_PATH (260) символами, это может быть относительный путь или только имя файла, функция VFP [FULLPATH](#) используется для получения полного имени файла. Используется обычный путь поиска VFP ([SET PATH](#)).

Возвращаемое значение

Длинный путь к указанному файлу.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[GetLongPathName](#)

GetOpenFileNameEx

Создает диалоговое окно «Открыть», в котором пользователь может указать диск, каталог и имя файла или набора файлов, которые необходимо открыть.

```
GetOpenFileNameEx([ nFlags [, cFileFilters [, cFileName [,
cInitialDirectory [, cDialogTitle [, nFlagsEx [, cArrayName [,
cCallbackFunc ]]]]]]])
```

Параметры

nFlags (необязательный, дополнительный)

если параметр *nFlags* опущен или передан 0, используются следующие флаги:
 OFN_EXPLORER | OFN_NOCHANGEDIR | OFN_NODEREFERENCELINKS |
 OFN_FILEMUSTEXIST | OFN_DONTADDTORECENT

Допустимые флаги:

Флаг	Значение
OFN_CREATEPROMPT	Если пользователь указывает файл, который не существует, этот флаг заставляет диалоговое окно запрашивать у пользователя разрешение на создание файла.
OFN_DONTADDTORECENT	Windows 2000/XP: запрещает системе добавлять ссылку на выбранный файл в каталоге файловой системы, содержащем последние использованные пользователем документы.
OFN_EXPLORER	Если вам нужен старый пользовательский интерфейс, опустите флаг OFN_EXPLORER.
OFN_FILEMUSTEXIST	Указывает, что пользователь может вводить только имена существующих файлов в поле ввода имени файла. Если этот флаг указан и пользователь вводит недопустимое имя, процедура диалогового окна отображает предупреждение в окне сообщения. Если этот флаг указан, также используется флаг OFN_PATHMUSTEXIST. Этот флаг можно использовать в диалоговом окне «Открыть». Его нельзя использовать с диалоговым окном «Сохранить как».
OFN_FORCESHOWHIDDEN	Windows 2000/XP: принудительно отображает системные и скрытые файлы, тем самым переопределяя пользовательскую настройку отображения или не отображения скрытых файлов. Однако файл, помеченный как системный и скрытый, не отображается.
OFN_LONGNAMES	Для диалоговых окон старого стиля этот флаг заставляет диалоговое окно использовать длинные имена файлов. Если этот флаг не указан или если также установлен флаг OFN_ALLOWMULTISELECT, диалоговые окна старого стиля используют короткие имена файлов (формат 8.3) для имен файлов с пробелами. Диалоговые окна в стиле проводника игнорируют этот флаг и всегда отображают длинные имена файлов.
OFN_NOCHANGEDIR	Восстанавливает текущий каталог до его исходного значения, если пользователь изменил каталог во время поиска файлов. Windows NT 4.0/2000/XP: Этот флаг неэффективен для GetOpenFileName .

OFN_NODEREFERENCELINKS	Указывает диалоговому окну вернуть путь и имя файла выбранного файла ярлыка (.LNK). Если это значение не указано, диалоговое окно возвращает путь и имя файла, на который ссылается ярлык.
OFN_NOLONGNAMES	Для диалоговых окон старого стиля этот флаг заставляет диалоговое окно использовать короткие имена файлов (формат 8.3). Диалоговые окна в стиле проводника игнорируют этот флаг и всегда отображают длинные имена файлов.
OFN_NONETWORKBUTTON	Скрывает и отключает кнопку «Сеть».
OFN_NOREADONLYRETURN	Указывает, что для возвращаемого файла не установлен флажок «Только чтение» и он не находится в каталоге, защищенном от записи.
OFN_NOTESTFILECREATE	Указывает, что файл не создается до закрытия диалогового окна. Этот флаг следует указать, если приложение сохраняет файл в сетевом ресурсе create-notmodify. Когда приложение указывает этот флаг, библиотека не проверяет защиту от записи, заполненный диск, открытую дверцу привода или сетевую защиту. Приложения, использующие этот флаг, должны выполнять файловые операции осторожно, поскольку файл нельзя открыть повторно после его закрытия.
OFN_OVERWRITEPROMPT	Заставляет диалоговое окно «Сохранить как» генерировать окно сообщения, если выбранный файл уже существует. Пользователь должен подтвердить, следует ли перезаписывать файл.
OFN_PATHMUSTEXIST	Указывает, что пользователь может вводить только допустимые пути и имена файлов. Если этот флаг используется и пользователь вводит недопустимый путь и имя файла в поле ввода имени файла, функция диалогового окна отображает предупреждение в окне сообщения.
OFN_READONLY	Заставляет флажок Только для чтения быть изначально выбранным при создании диалогового окна. Этот флаг указывает состояние флажка Только для чтения, когда диалоговое окно закрыто.

cFileFilters (необязательно)

по умолчанию = «Все» + CHR(0) + «*.*»

CHR(0) разделяет символьные строки и фильтры, например: «Таблицы» + CHR(0) + «*.dbf» + CHR(0) + «Базы данных» + CHR(0) + «*.dbc».

Вы также можете передать "" (пустую строку), чтобы пропустить значения фильтра.

cFileName (необязательно)

по умолчанию = пусто

Имя файла, которое изначально отображается в диалоговом окне.

cInitialDirectory (необязательно)

по умолчанию = пусто

Начальный каталог. Алгоритм выбора начального каталога различается на разных платформах.

Windows 7: если cInitialDirectory имеет то же значение, что было передано при первом

использовании приложением диалогового окна «Открыть» или «Сохранить как», в качестве начального каталога используется последний выбранный пользователем путь. В противном случае, если *cFileName* содержит путь, этот путь является начальным каталогом.

В противном случае, если *cInitialDirectory* не пуст, он указывает начальный каталог. Если *cInitialDirectory* пуст и текущий каталог содержит какие-либо файлы указанных типов фильтров, начальным каталогом является текущий каталог.

В противном случае начальным каталогом является каталог личных файлов текущего пользователя.

В противном случае начальным каталогом является папка «Рабочий стол».

Windows 2000/XP/Vista: если *cFileName* содержит путь, этот путь является начальным каталогом.

В противном случае *cInitialDirectory* указывает начальный каталог.

В противном случае, если приложение использовало диалоговое окно «Открыть» или «Сохранить как» в прошлом, в качестве начального каталога выбирается последний использованный путь. Однако если приложение не запускается в течение длительного времени, его сохраненный выбранный путь отбрасывается.

Если *cInitialDirectory* пуст и текущий каталог содержит какие-либо файлы указанных типов фильтров, то начальным каталогом является текущий каталог.

В противном случае начальным каталогом является каталог личных файлов текущего пользователя.

В противном случае начальным каталогом является папка Desktop.

***cDialogTitle* (необязательно)**

по умолчанию = пусто

Строка, которая будет помещена в заголовок диалогового окна. Если этот параметр пуст, система использует заголовок по умолчанию (то есть Save As или Open).

***nFlagsEx* (необязательно)**

по умолчанию = 0,

если вы передадите OFN_EX_NOPLACESBAR (1) в качестве параметра *nFlagsEx*, диалоговое окно не будет отображать панель мест (кнопка слева для часто используемых каталогов).

***cArrayName* (необязательно)**

Если вы передадите *arrayname*, диалоговое окно позволит выбрать несколько файлов. Вместо имени файла функция вернет количество выбранных файлов. Первый элемент массива будет содержать путь к выбранным файлам, каждый последующий элемент будет содержать имя файла (без пути).

***cCallbackfunction* (необязательно)**

Если вы передадите имя функции VFP, функция будет вызвана при возникновении событий в диалоге.

Функция должна иметь следующий прототип.

```
FUNCTION OpenFileDialogCallback(lnHwnd, lnControlID, lnCode)
ENDFUNC
```

Для получения дополнительной информации см. `dialogs.prg` в примере проекта.

Возвращаемое значение

-1, если произошла ошибка.

0, если диалог был прерван.

Выбранное имя файла для диалога с одиночным выбором или количество выбранных файлов, если для множественного выбора было передано имя массива.

Смотрите также

Ссылки[GetSaveFileNameEx](#)[MessageBoxEx](#)[SHBrowseFolder](#)**Используемые функции WinApi**[GetOpenFileName](#)[CommDlgExtendedError](#)

GetSaveFileNameEx

Создает диалоговое окно «Сохранить», в котором пользователь может указать диск, каталог и имя файла для сохранения.

```
GetSaveFileNameEx([ nFlags [, cFileFilters [, cFileName [,
cInitialDirectory [, cDialogTitle [, nFlagsEx [,
cCallbackFunc ]]]]]])
```

Параметры

nFlags (необязательный, дополнительный)

Если параметр nFlags опущен или передан 0, используются следующие флаги:
OFN_EXPLORER | OFN_NOCHANGEDIR

Допустимые флаги:

Флаг	Значение
OFN_CREATEPROMPT	Если пользователь указывает файл, который не существует, этот флаг заставляет диалоговое окно запрашивать у пользователя разрешение на создание файла.
OFN_DONTADDTORECENT	Windows 2000/XP: запрещает системе добавлять ссылку на выбранный файл в каталоге файловой системы, содержащем последние использованные пользователем документы.
OFN_EXPLORER	Если вам нужен старый пользовательский интерфейс, опустите флаг OFN_EXPLORER.
OFN_FILEMUSTEXIST	Указывает, что пользователь может вводить только имена существующих файлов в поле ввода имени файла. Если этот флаг указан и пользователь вводит недопустимое имя, процедура диалогового окна отображает предупреждение в окне сообщения. Если этот флаг указан, также используется флаг OFN_PATHMUSTEXIST. Этот флаг можно использовать в диалоговом окне «Открыть». Его нельзя использовать с диалоговым окном «Сохранить как».
OFN_FORCESHOWHIDDEN	Windows 2000/XP: принудительно отображает системные и скрытые файлы, тем самым переопределяя пользовательскую настройку отображения или не отображения скрытых файлов. Однако файл, помеченный как системный и скрытый, не отображается.
OFN_LONGNAMES	Для диалоговых окон старого стиля этот флаг заставляет диалоговое окно использовать длинные имена файлов. Если этот флаг не указан или если также установлен флаг OFN_ALLOWMULTISELECT, диалоговые окна старого стиля используют короткие имена файлов (формат 8.3) для имен файлов с пробелами. Диалоговые окна в стиле проводника игнорируют этот флаг и всегда отображают длинные имена файлов.
OFN_NOCHANGEDIR	Восстанавливает текущий каталог до исходного значения, если пользователь изменил каталог во время поиска файлов.
OFN_NODEREFERENCELINKS	Указывает диалоговому окну вернуть путь и имя файла выбранного файла ярлыка (.LNK). Если это значение не указано, диалоговое окно возвращает путь и имя

	файла, на который ссылается ярлык.
OFN_NOLONGNAMES	Для диалоговых окон старого стиля этот флаг заставляет диалоговое окно использовать короткие имена файлов (формат 8.3). Диалоговые окна в стиле проводника игнорируют этот флаг и всегда отображают длинные имена файлов.
OFN_NONETWORKBUTTON	Скрывает и отключает кнопку «Сеть».
OFN_NOREADONLYRETURN	Указывает, что для возвращаемого файла не установлен флажок «Только чтение» и он не находится в каталоге, защищенном от записи.
OFN_NOTESTFILECREATE	Указывает, что файл не создается до закрытия диалогового окна. Этот флаг следует указать, если приложение сохраняет файл в сетевом ресурсе create-on-modify. Когда приложение указывает этот флаг, библиотека не проверяет защиту от записи, заполненный диск, открытую дверцу привода или сетевую защиту. Приложения, использующие этот флаг, должны выполнять файловые операции осторожно, поскольку файл нельзя открыть повторно после его закрытия.
OFN_OVERWRITEPROMPT	Заставляет диалоговое окно «Сохранить как» генерировать окно сообщения, если выбранный файл уже существует. Пользователь должен подтвердить, следует ли перезаписывать файл.
OFN_PATHMUSTEXIST	Указывает, что пользователь может вводить только допустимые пути и имена файлов. Если этот флаг используется и пользователь вводит недопустимый путь и имя файла в поле ввода имени файла, функция диалогового окна отображает предупреждение в окне сообщения.
OFN_READONLY	Заставляет флажок Только для чтения быть изначально выбранным при создании диалогового окна. Этот флаг указывает состояние флажка Только для чтения, когда диалоговое окно закрыто.

cFileFilters (необязательно)

по умолчанию = «Все» + CHR(0) + «*.*»

CHR(0) разделяет символьные строки и фильтры, например: «Таблицы» + CHR(0) + «*.dbf» + CHR(0) + «Базы данных» + CHR(0) + «*.dbc».

Вы также можете передать "" (пустую строку), чтобы пропустить значения фильтра.

cFileName (необязательно)

по умолчанию = пусто

Имя файла, которое изначально отображается в диалоговом окне.

cInitialDirectory (необязательно)

по умолчанию = пусто

Начальный каталог. Алгоритм выбора начального каталога различается на разных платформах.

Windows 7: если cInitialDirectory имеет то же значение, что было передано при первом использовании приложением диалогового окна «Открыть» или «Сохранить как», в качестве начального каталога используется последний выбранный пользователем путь. В противном случае, если cFileName содержит путь, этот путь является начальным

каталогом.

В противном случае, если *cInitialDirectory* не пуст, он указывает начальный каталог. Если *cInitialDirectory* пуст и текущий каталог содержит какие-либо файлы указанных типов фильтров, начальным каталогом является текущий каталог.

В противном случае начальным каталогом является каталог личных файлов текущего пользователя. В противном случае начальным каталогом является папка «Рабочий стол».

Windows 2000/XP/Vista: если *cFileName* содержит путь, этот путь является начальным каталогом.

В противном случае *cInitialDirectory* указывает начальный каталог.

В противном случае, если приложение использовало диалоговое окно «Открыть» или «Сохранить как» в прошлом, в качестве начального каталога выбирается последний использованный путь. Однако если приложение не запускается в течение длительного времени, его сохраненный выбранный путь отбрасывается. Если *cInitialDirectory* пуст и текущий каталог содержит какие-либо файлы указанных типов фильтров, то начальным каталогом является текущий каталог.

В противном случае начальным каталогом является каталог личных файлов текущего пользователя. В противном случае начальным каталогом является папка Desktop.

***cDialogTitle* (необязательно)**

по умолчанию = пусто

Строка, которая будет помещена в заголовок диалогового окна. Если этот параметр пуст, система использует заголовок по умолчанию (то есть Save As или Open).

***nFlagsEx* (необязательно)**

по умолчанию = 0,

если вы передадите OFN_EX_NOPLACESBAR (1) в качестве параметра *nFlagsEx*, диалоговое окно не будет отображать панель мест (кнопка слева для часто используемых каталогов).

***cCallbackfunction* (необязательно)**

Если вы передадите имя функции VFP, функция будет вызвана при возникновении событий в диалоге.

Функция должна иметь следующий прототип.

```
FUNCTION OpenFileDialogCallback(lnHwnd, lnControlID, lnCode)
ENDFUNC
```

Для получения дополнительной информации см. dialogs.prg в примере проекта.

Возвращаемое значение

-1, если произошла ошибка.

0, если диалог был прерван.

Введенное имя файла.

Смотрите также

Ссылки

[GetOpenFileNameEx](#)

[MessageBoxEx](#)

[SHBrowseFolder](#)

Используемые функции WinApi

[GetSaveFileName](#)

[CommDlgExtendedError](#)

GetServerTime

Извлекает текущее время с сервера Windows.

```
GetServerTime( cServer [, nTimeZone ] )
```

Параметры

cServer

DNS- или NetBIOS-имя удаленного сервера, с которого необходимо получить время.

Примечание

Windows NT: эта строка должна начинаться с \\.

nTimeZone (необязательно)

по умолчанию = 1

Часовой пояс, в который должно быть преобразовано возвращаемое время.

Одно из следующих значений:

Часовой пояс	Значение
1	универсальное глобальное время
2	Часовой пояс локального компьютера.
3	Часовой пояс сервера.

Возвращаемое значение

Значение даты и времени.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)

[AIPAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[ICMPing](#)

[Ip2MacAddress](#)

[Resolve HostToIp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

Используемые функции WinApi

[NetRemoteTOD](#)

GetShortPathNameEx

Возвращает краткую форму указанного пути.

```
GetShortPathNameEx( cFileName )
```

Параметры

cFileName

Путь, который необходимо преобразовать.

Примечание

Имя ограничено MAX_PATH (260) символами, это может быть относительный путь или только имя файла, функция VFP [FULLPATH](#) используется для получения полного имени файла. Используется обычный путь поиска VFP ([SET PATH](#)).

Возвращаемое значение

Короткий путь к указанному файлу.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[GetShortPathName](#)

GetSystemDirectoryEx

Возвращает путь к системному каталогу Windows.

```
GetSystemDirectoryEx()
```

Возвращаемое значение

Системный каталог.

Смотрите также

Ссылки

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStringsEx](#)

[GetLocaleInfoEx](#)

[GetWindowsDirectoryEx](#)

[OsEx](#)

Используемые функции WinApi

[GetSystemDirectory](#)

GetSystemTimeEx

Возвращает текущую системную дату и время по всемирному координированному времени (UTC) или по местному времени.

```
GetSystemTimeEx([ bUTCTime ])
```

Параметры

bUTCTime (необязательно)

по умолчанию = .F.

Если .T. время возвращается в формате UTC.

Возвращаемое значение

Значение даты и времени.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

Используемые функции WinApi

[GetLocalTime](#)

[GetSystemTime](#)

GetWindowRectEx

Возвращает размеры ограничивающего прямоугольника указанного окна.

```
GetWindowRectEx( hWnd , cArrayName )
```

Параметры

hWnd

Дескриптор окна.

cArrayName

По возвращении массив содержит следующие значения из структуры [RECT](#).

Элемент	Значение
1	Left
2	Right
3	Top
4	Bottom

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AWindowProps](#)

[AWindows](#)

[AWindowsEx](#)

[CenterWindowEx](#)

[GetWindowTextEx](#)

Используемые функции WinApi

[GetWindowRect](#)

GetWindowsDirectoryEx

Возвращает путь к каталогу Windows.

```
GetWindowsDirectoryEx()
```

Возвращаемое значение

Каталог Windows.

Смотрите также

Ссылки

[ADesktopArea](#)

[ADesktops](#)

[ADisplayDevices](#)

[AResolutions](#)

[AWindowStations](#)

[ExpandEnvironmentStringsEx](#)

[GetLocaleInfoEx](#)

[GetSystemDirectoryEx](#)

[OsEx](#)

Используемые функции WinApi

[GetWindowsDirectory](#)

GetWindowTextEx

Извлекает текст, соответствующий окну.

```
GetWindowTextEx( hWnd )
```

Параметры

hWnd

Дескриптор окна.

Возвращаемое значение

Заголовок окна.

Смотрите также

Ссылки

[AWindowProps](#)

[AWindows](#)

[AWindowsEx](#)

[CenterWindowEx](#)

[GetWindowRectEx](#)

Используемые функции WinApi

[SendMessageTimeout](#)

ICMPing

Отправляет эхо-запрос IPv4 ICMP, также известный как ping, и возвращает любые эхо-ответы.

```
IcmpPing( cArrayName , cHost [, nTTL [, nTOS [, nTimeout [,  
nDatasize [, bDontFragment [, nPingCount ]]]]])
```

Параметры

cArrayname

При возврате массив содержит следующую информацию:

Если пинг прошел успешно

Столбец	Значение	Тип данных
1	обратный IP-адрес	C
2	Время передачи туда и обратно в миллисекундах	N
3	Статус	N
4	Данные были успешно извлечены обратно	L

Если пинг не удался

Столбец	Значение
1	пустая строка
2	-1
3	-1
4	.F.

cHost

Либо IP-адрес, либо имя хоста.

Например, «192.168.1.128», «www.google.com»

nTTL (необязательно)

по умолчанию = 30

Время жизни сетевого пакета.

nTOS (необязательно)

по умолчанию = 0

Тип услуги.

nTimeout (необязательно)

по умолчанию = 3000 мс

Время ожидания в миллисекундах.

nDdatasize (необязательно)

по умолчанию = 32 (байта)

Количество отправляемых байтов.

bDontFragment (необязательно)

по умолчанию = .F.

Может ли пакет быть фрагментирован?

nPingCount (необязательно)

по умолчанию = 1

Количество отправляемых пингов, каждый результат пинга будет сохранен в новой строке массива.

Возвращаемое значение

Количество выполненных пингов.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)

[AIPAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[Ip2MacAddress](#)

[ResolveHostToIp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

Используемые функции WinApi

[IcmpCreateFile](#)

[IcmpSendEcho](#)

[IcmpCloseHandle](#)

[gethostbyname](#)

Int64_Add

Складывает 64-битные целые числа.

```
Int64_Add( ycqnBigInt , ycqnBigInt2 [, nFormat ])
```

Параметры

ycqnBigInt

Первый операнд сложения.

Параметры могут быть переданы в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "-1234567890123"
4. Как обычное числовое значение VFP.

ycqnBigInt2

Второй операнд сложения.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	Строковый литерал.
3	8-байтовая двоичная строка.

Возвращаемое значение

Результат сложения в запрошенном формате.

Замечания

Функция выдает ошибку 39 "Числовое переполнение. Данные были утеряны." если результат превышает предел 64-значного целого числа со знаком, от -9223372036854775808 до 9223372036854775807.

Значение валюты, возвращаемое функцией, должно использоваться только со следующими функциями: [Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int642Str](#), [WriteInt64](#), [WritePInt64](#) и [WriteRegistryKey](#).

Тип данных валюты VFP имеет подразумеваемую десятичную точку. При конвертации значения валюты, возвращаемого из [Int_Add](#), [Int64_Sub](#) и т. д. с помощью [MTON\(\)](#), возвращаемое числовое значение не соответствует ожидаемому, кроме того, каждая функция VFP выдаст ошибку 1988, если значение равно -9 223 372 036 854 775 808.

Поскольку эта функция может возвращать недопустимое значение валюты (-9 223 372 036 854 775 808), следует использовать тип данных Q(8) вместо Y для хранения 64-битных целых чисел в таблице.

Пример

```
? Int64_Add("-9223372036854775808", 8, 2)
```

Смотрите также

Ссылки

[AFontInfo](#)
[ASplitStr](#)
[Decimals](#)
[GetCursorPosEx](#)
[Int64_Div](#)
[Int64_Mod](#)
[Int64_Mul](#)
[Int64_Sub](#)

Int64_Div

Делит 64-битные целые числа.

```
Int64_Div( yqcnBigInt , yqcnBigInt2 [, nFormat ])
```

Параметры

yqcnBigInt

Первый операнд деления.

Параметры могут быть переданы в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая varbinary-строка.
3. Как строковый литерал, например "-1234567890123"
4. Как обычное числовое значение VFP.

yqcnBigInt2

Второй операнд деления.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	Строковый литерал.
3	8-байтовая двоичная строка.

Возвращаемое значение

Результат деления в запрошенном формате.

Замечания

Функция выдает ошибку 1307 "Невозможно разделить на 0." если вы передаете 0 в качестве делителя.

Функция выдает ошибку 39 "Числовое переполнение. Данные были утеряны." при делении -9223372036854775808 на -1, так как +9223372036854775808 не входит в представимый диапазон 64-битного целого числа со знаком.

Возвращаемое функцией значение валюты следует использовать только со следующими функциями:

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int642Str](#), [WriteInt64](#), [WritePInt64](#) и [WriteRegistryKey](#).

Тип данных валюты VFP имеет подразумеваемую десятичную точку. При конвертации значения валюты, возвращаемого из [Int_Add](#), [Int64_Sub](#) и т. д. с помощью [MTON\(\)](#), возвращаемое числовое значение не соответствует ожидаемому, кроме того, каждая функция VFP выдаст ошибку 1988, если значение равно -9 223 372 036 854 775 808.

Поскольку эта функция может возвращать недопустимое значение валюты (-9 223 372 036 854 775 808), следует использовать тип данных Q(8) вместо Y для хранения 64-битных целых чисел в таблице.

Смотрите также

Ссылки[AFontInfo](#)[ASplitStr](#)[Decimals](#)[GetCursorPosEx](#)[Int64_Add](#)[Int64_Mod](#)[Int64_Mul](#)[Int64_Sub](#)

Int64_Mod

Делит одно 64-битное целое число на другое и возвращает остаток.

```
Int64_Mod( yqcnBigInt , yqcnBigInt2 [, nFormat ])
```

Параметры

yqcnBigInt

Делимое.

Параметры могут быть переданы в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая varbinary строка.
3. Как строковый литерал, например "-1234567890123"
4. Как обычное числовое значение VFP.

yqcnBigInt2

Делитель.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	Строковый литерал.
3	8-байтовая двоичная строка.

Возвращаемое значение

Результат операции по модулю в запрошенном формате.

Замечания

Функция вызывает ошибку 1307 «Невозможно разделить на 0.», если вы передаете 0 в качестве делителя.

Возвращаемое функцией значение валюты должно использоваться только со следующими функциями:

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int642Str](#), [WriteInt64](#), [WritePInt64](#) и [WriteRegistryKey](#).

Тип данных валюты VFP имеет подразумеваемую десятичную точку. При преобразовании значения валюты, возвращаемого из [Int_Add](#), [Int64_Sub](#) и т. д. с помощью [MTON\(\)](#), возвращаемое числовое значение не соответствует ожидаемому, кроме того, каждая функция VFP вызовет ошибку 1988, если значение равно -9 223 372 036 854 775 808.

Смотрите также

Ссылки

[AFontInfo](#)
[ASplitStr](#)
[Decimals](#)
[GetCursorPosEx](#)
[Int64_Add](#)
[Int64_Div](#)

[Int64_Mul](#)[Int64_Sub](#)

Int64_Mul

Умножает 64-битные целые числа.

```
Int64_Mul( yqcnBigInt , yqcnBigInt2 [, nFormat ])
```

Параметры

yqcnBigInt

Первый операнд умножения.

Параметры могут быть переданы в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "-1234567890123"
4. Как обычное числовое значение VFP.

yqcnBigInt2

Второй операнд умножения.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	Строковый литерал.
3	8-байтовая двоичная строка.

Возвращаемое значение

Результат умножения в запрошенном формате.

Замечания

Функция выдает ошибку 39 "Числовое переполнение. Данные были утеряны." если результат превышает предел 64-значного целого числа со знаком, от -9223372036854775808 до 9223372036854775807.

Значение валюты, возвращаемое функцией, следует использовать только со следующими функциями:

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int642Str](#), [WriteInt64](#), [WritePInt64](#) и [WriteRegistryKey](#).

Тип данных валюты VFP имеет подразумеваемую десятичную точку. При конвертации значения валюты, возвращаемого из [Int_Add](#), [Int64_Sub](#) и т. д. с помощью [MTON\(\)](#), возвращаемое числовое значение не соответствует ожидаемому, кроме того, каждая функция VFP выдаст ошибку 1988, если значение равно -9 223 372 036 854 775 808.

Поскольку эта функция может возвращать недопустимое значение валюты (-9 223 372 036 854 775 808), следует использовать тип данных Q(8) вместо Y для хранения 64-битных целых чисел в таблице.

Смотрите также

Ссылки

[AFontInfo](#)

[ASplitStr](#)

[Decimals](#)
[GetCursorPosEx](#)
[Int64_Add](#)
[Int64_Div](#)
[Int64_Mod](#)
[Int64_Sub](#)

Int64_Sub

Вычитает 64-битные целые числа.

```
Int64_Sub( yqcnBigInt , yqcnBigInt2 [, nFormat ])
```

Параметры

yqcnBigInt

Первый операнд вычитания.

Параметры могут быть переданы в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "-1234567890123"
4. Как обычное числовое значение VFP.

yqcnBigInt2

Второй операнд вычитания.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	Строковый литерал.
3	8-байтовая двоичная строка.

Возвращаемое значение

Результат вычитания в требуемом формате.

Замечания

Функция выдает ошибку 39 "Числовое переполнение. Данные были утеряны." если результат превышает предел 64-значного целого числа со знаком, от -9223372036854775808 до 9223372036854775807.

Значение валюты, возвращаемое функцией, следует использовать только со следующими функциями:

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int642Str](#), [WriteInt64](#), [WritePInt64](#) и [WriteRegistryKey](#).

Тип данных валюты VFP имеет подразумеваемую десятичную точку. При конвертации значения валюты, возвращаемого из [Int_Add](#), [Int64_Sub](#) и т. д. с помощью [MTON\(\)](#), возвращаемое числовое значение не соответствует ожидаемому, кроме того, каждая функция VFP выдаст ошибку 1988, если значение равно -9 223 372 036 854 775 808.

Поскольку эта функция может возвращать недопустимое значение валюты (-9 223 372 036 854 775 808), следует использовать тип данных Q(8) вместо Y для хранения 64-битных целых чисел в таблице.

Смотрите также

Ссылки

[AFontInfo](#)

[ASplitStr](#)

[Decimals](#)
[GetCursorPosEx](#)
[Int64_Add](#)
[Int64_Div](#)
[Int64_Mod](#)
[Int64_Mul](#)

Int642Str

Преобразует значение __int64 (64-битное числовое значение) в требуемый формат.

```
Int642Str( yqcnValue [, nFormat ])
```

Параметры

yqcnValue

Числовое значение в диапазоне от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807.

Параметры могут быть переданы в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "-1234567890123"
4. Как обычное числовое значение VFP.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	8-байтовая двоичная строка.
2	Строковый литерал.

Возвращаемое значение

Значение __int64 в запрошенном формате.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Ip2MacAddress

Возвращает MAC-адрес для заданного IP-адреса.

```
Ip2MacAddress( cIP )
```

Параметры

cIP

IP-адрес, для которого необходимо получить MAC-адрес.

Возвращаемое значение

MAC-адрес (строка) переданного IP-адреса.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)

[AIPAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[ICmpPing](#)

[Resolve HostToIp](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

Используемые функции WinApi

[SendARP](#)

[inet_addr](#)

IsEqualGuid

Сравнивает два GUID на предмет эквивалентности.

```
IsEqualGuid( cGuid1Binary | cGuid1String | nGuid1Pointer ,  
cGuid2Binary | cGuid2String | nGuid2Pointer )
```

Параметры

Возвращаемое значение

.Т. если GUID равны, .F. в противном случае.

Смотрите также

Ссылки

[CLSIDFromProgID](#)
[CLSIDFromString](#)
[CreateGuid](#)
[CreateThreadObject](#)
[GetIUnknown](#)
[ProgIDFromCLSID](#)
[RegisterActiveObject](#)
[RegisterObjectAsFileMoniker](#)
[RevokeActiveObject](#)
[StringFromCLSID](#)

Используемые функции WinApi

[CLSIDFromString](#)

LockHGlobal

Блокирует глобальный объект памяти и возвращает указатель на первый байт блока памяти объекта.

```
LockHGlobal ( nHandle )
```

Параметры

nHandle

Дескриптор блока памяти, возвращаемый [AllocHGlobal](#).

Примечание

Дескриптор должен быть выделен с флагом GMEM_MOVEABLE.

Возвращаемое значение

Указатель (числовой) на заблокированную область памяти.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[GlobalLock](#)

Long2Str

Преобразует длинное знаковое числовое значение (32-битное число) в двоичную строку.

```
Long2Str( nValue )
```

Параметры

nValue

Числовое значение в диапазоне от -2 147 483 648 до 2 147 483 647.

Возвращаемое значение

Строка, которая в двоичном виде равна переданному длинному значению.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

MarshalCArray2Cursor

Преобразует массив C в курсор VFP.

```
MarshalCArray2Cursor( nAddress , cCursorAndFieldNames , nType ,
nRows [, nLength | nCodePage [, nCodePage]])
```

Параметры

nAddress

Базовый адрес массива C.

cCursorAndFieldNames

Имена курсора и полей, например, «cursorname.fieldname, fieldname2».

nType

Тип данных C массива.

Одно из следующих значений.

Тип	Декларация массива C
CTYPE_SHORT	short array[]
CTYPE_USHORT	unsigned short array[]
CTYPE_INT	int array[]
CTYPE_UINT	unsigned int array[]
CTYPE_FLOAT	float array[]
CTYPE_DOUBLE	double array[]
CTYPE_BOOL	BOOL array[]
CTYPE_CSTRING	char* array[]
CTYPE_WSTRING	wchar_t* array[]
CTYPE_CHARARRAY	char array[][]
CTYPE_WCHARARRAY	wchar_t array[][]
CTYPE_INT64	__int64 array[]
CTYPE_UINT64	unsigned __int64 array[]

nRows

Количество строк для упорядочивания.

nLength | nCodePage (необязательно)

Либо длина массива символов, либо кодовая страница, используемая для преобразования Unicode в ANSI.

nCodePage (необязательно)

Кодовая страница, используемая для преобразования Unicode в ANSI.

Возвращаемое значение

Всегда .T.

Замечания

Для массивов указателей символьного типа CTYPE_CSTRING и CTYPE_WSTRING указатель 0 преобразуется в значение NULL.

Смотрите также

Ссылки

[MarshalCArray2FoxArray](#)

[MarshalCursor2CArray](#)

[MarshalFoxArray2CArray](#)

MarshalCArray2FoxArray

Преобразует массив C в массив VFP.

```
MarshalCArray2FoxArray( nAddress , @aArray , nType [, nLength |
nCodePage [, nCodePage]])
```

Параметры

nAddress

Базовый адрес массива C.

@aArray

Целевой массив VFP по ссылке.

Размеры массива указывают размер массива C.

Примечание

Количество строк и столбцов ограничено 65000.

nType

Тип данных C массива.

Одно из следующих значений.

Тип	Декларация массива C
CTYPE_SHORT	short array[]
CTYPE_USHORT	unsigned short array[]
CTYPE_INT	int array[]
CTYPE_UINT	unsigned int array[]
CTYPE_FLOAT	float array[]
CTYPE_DOUBLE	double array[]
CTYPE_BOOL	BOOL array[]
CTYPE_CSTRING	char* array[]
CTYPE_WSTRING	wchar_t* array[]
CTYPE_CHARARRAY	char array[][]
CTYPE_WCHARARRAY	wchar_t array[][]
CTYPE_INT64	__int64 array[]
CTYPE_UINT64	unsigned __int64 array[]

nLength | nCodePage (необязательно)

Либо длина массива символов, либо кодовая страница, используемая для преобразования Unicode в ANSI.

nCodePage (необязательно)

Кодовая страница, используемая для преобразования Unicode в ANSI.

Возвращаемое значение

Всегда .T.

Замечания

Для массивов указателей символьного типа CTYPE_CSTRING и CTYPE_WSTRING указатель 0 преобразуется в значение NULL.

Смотрите также

Ссылки

[MarshalCArray2Cursor](#)
[MarshalCursor2CArray](#)
[MarshalFoxArray2CArray](#)

MarshalCursor2CArray

Преобразует курсор VFP в массив C.

```
MarshalCursor2CArray( nAddress , cCursorAndFieldNames , nType [,
nLength | nCodePage [, nCodePage]])
```

Параметры

nAddress

Базовый адрес массива C.

cCursorAndFieldNames

Имена курсора и полей, например, «cursorname.fieldname, fieldname2».

nType

Тип данных C массива.

Одно из следующих значений.

Тип	Декларация массива C
CTYPE_SHORT	short array[]
CTYPE_USHORT	unsigned short array[]
CTYPE_INT	int array[]
CTYPE_UINT	unsigned int array[]
CTYPE_FLOAT	float array[]
CTYPE_DOUBLE	double array[]
CTYPE_BOOL	BOOL array[]
CTYPE_CSTRING	char* array[]
CTYPE_WSTRING	wchar_t* array[]
CTYPE_CHARARRAY	char array[][]
CTYPE_WCHARARRAY	wchar_t array[][]
CTYPE_INT64	__int64 array[]
CTYPE_UINT64	unsigned __int64 array[]

nLength | nCodePage (необязательно)

Либо длина массива символов, либо кодовая страница, используемая для преобразования ANSI в Unicode.

nCodePage (необязательно)

Кодовая страница, используемая для преобразования ANSI в Unicode.

Возвращаемое значение

Всегда .T.

Замечания

Поля курсора должны быть правильного типа или NULL, в противном случае возникнет ошибка "недопустимые аргументы".

Значения NULL игнорируются, а значение в текущем индексе массива не изменяется. Для массивов указателей символьного типа CTYPE_CSTRING и CTYPE_WSTRING значение NULL приводит к нулевому указателю.

Смотрите также

Ссылки

[MarshalCArray2Cursor](#)

[MarshalCArray2FoxArray](#)

[MarshalFoxArray2CArray](#)

MarshalFoxArray2CArray

Преобразует массив VFP в массив C.

```
MarshalFoxArray2CArray( nAddress , @aArray , nType [, nLength |
nCodePage [, nCodePage]])
```

Параметры

nAddress

Базовый адрес массива C.

@aArray

Массив VFP для преобразования по ссылке.

nType

Тип данных C массива.

Одно из следующих значений.

Тип	Декларация массива C
CTYPE_SHORT	short array[]
CTYPE_USHORT	unsigned short array[]
CTYPE_INT	int array[]
CTYPE_UINT	unsigned int array[]
CTYPE_FLOAT	float array[]
CTYPE_DOUBLE	double array[]
CTYPE_BOOL	BOOL array[]
CTYPE_CSTRING	char* array[]
CTYPE_WSTRING	wchar_t* array[]
CTYPE_CHARARRAY	char array[][]
CTYPE_WCHARARRAY	wchar_t array[][]
CTYPE_INT64	__int64 array[]
CTYPE_UINT64	unsigned __int64 array[]

nLength | nCodePage (необязательно)

Либо длина массива символов, либо кодовая страница, используемая для преобразования ANSI в Unicode.

nCodePage (необязательно)

Кодовая страница, используемая для преобразования ANSI в Unicode.

Возвращаемое значение

Всегда .T.

Замечания

Элементы массива должны быть правильного типа или NULL, в противном случае возникает ошибка "недопустимые аргументы".

Значения NULL игнорируются, а значение в текущем индексе массива не изменяется. Для массивов указателей символьного типа CTYPE_CSTRING и CTYPE_WSTRING значение NULL приводит к нулевому указателю.

Смотрите также

Ссылки

[MarshalCArray2Cursor](#)

[MarshalCArray2FoxArray](#)

[MarshalCursor2CArray](#)

MessageBoxEx

Создает, отображает и управляет окном сообщения. Окно сообщения содержит текст и заголовок сообщения, определяемые приложением, любую иконку и любую комбинацию предопределенных кнопок.

```
MessageBoxEx( cText [, nFlags [, cCaption [, nHwnd [,
cnIconResource [, nHInstance [, nHelpContextId [,
nLanguageId ]]]]]])
```

Параметры

cText

Задаёт текст, который отображается в диалоговом окне.

Примечание

VFP [MESSAGEBOX](#) () позволяет передавать выражение любого типа в этом параметре - эта функция **НЕТ** !

Передаваемое значение должно быть строкой, просто используйте [TRANSFORM](#) () или другие функции форматирования, чтобы создать строку из выражения.

nFlags (необязательно)

по умолчанию = 0

Параметр *nFlags* управляет внешним видом и поведением окна сообщения.

Кнопки.

Значение	Описание
MB_OK	Только кнопка ОК
MB_OKCANCEL	Кнопки ОК и Отмена
MB_ABORTRETRYIGNORE	Кнопки «Отмена», «Повторить» и «Игнорировать»
MB_YESNOCANCEL	Кнопки «Да», «Нет» и «Отмена»
MB_YESNO	Кнопки «Да» и «Нет»
MB_RETRYCANCEL	Кнопки «Повторить» и «Отмена»

Значок.

Значение	Описание
MB_ICONSTOP	Значок останова
MB_ICONQUESTION	Вопросительный знак
MB_ICONEXCLAMATION	Восклицательный знак
MB_ICONINFORMATION	Значок Информации

Кнопка по умолчанию.

Значение	Описание
MB_DEFBUTTON1	Первая кнопка — это кнопка по умолчанию. MB_DEFBUTTON1 — кнопка по умолчанию, если не указано MB_DEFBUTTON2, MB_DEFBUTTON3 или MB_DEFBUTTON4.
MB_DEFBUTTON2	Вторая кнопка используется по умолчанию.
MB_DEFBUTTON3	Третья кнопка используется по умолчанию.

MB_DEFBUTTON4	Четвертая кнопка — кнопка по умолчанию.
---------------	---

Модальность диалогового окна.

Значение	Описание
MB_APPLMODAL	<p>Пользователь должен ответить на окно сообщения, прежде чем продолжить работу в окне, идентифицированном параметром <i>nWnd</i> . Однако пользователь может переходить в окна других потоков и работать в этих окнах.</p> <p>В зависимости от иерархии окон в приложении пользователь может иметь возможность переходить в другие окна в потоке. Все дочерние окна родителя окна сообщения автоматически отключаются, но всплывающие окна — нет.</p> <p>MB_APPLMODAL является значением по умолчанию, если не указаны ни MB_SYSTEMMODAL, ни MB_TASKMODAL.</p>
MB_SYSTEMMODAL	<p>То же, что и MB_APPLMODAL, за исключением того, что окно сообщения имеет стиль WS_EX_TOPMOST. Используйте системные модальные окна сообщений для уведомления пользователя о серьезных, потенциально опасных ошибках, требующих немедленного внимания (например, нехватка памяти). Этот флаг не влияет на способность пользователя взаимодействовать с окнами, отличными от тех, которые связаны с <i>nHwnd</i> .</p>
MB_TASKMODAL	<p>То же, что и MB_APPLMODAL, за исключением того, что все окна верхнего уровня, принадлежащие текущему потоку, отключаются, если параметр <i>nWnd</i> равен 0. Используйте этот флаг, когда вызывающее приложение или библиотека не имеет доступного дескриптора окна, но все равно должны запретить ввод в другие окна в вызывающем потоке без приостановки других потоков.</p>

Различные другие варианты.

Значение	Описание
MB_DEFAULT_DESKTOP_ONLY	<p>То же, что и рабочий стол интерактивной оконной станции. Для получения дополнительной информации см. Оконные станции.</p> <p>Если текущий входной рабочий стол не является рабочим столом по умолчанию, MessageBox не возвращается, пока пользователь не переключится на рабочий стол по умолчанию.</p>
MB_RIGHT	Текст выровнен по правому краю.
MB_RTLCREADIND	Отображает текст сообщения и подписи, используя порядок чтения справа налево в еврейских и арабских системах.
MB_SETFOREGROUND	Окно сообщения становится окном переднего плана. Внутри система вызывает функцию SetForegroundWindow для окна сообщения.
MB_TOPMOST	Окно сообщения создано с использованием стиля окна WS_EX_TOPMOST.
MB_SERVICE_NOTIFICATION	Вызывающий объект — это служба, уведомляющая

пользователя о событии. Функция отображает окно сообщения на текущем активном рабочем столе, даже если на компьютере нет пользователя, вошедшего в систему. Терминальные службы: если вызывающий поток имеет имперсонализирующий токен, функция направляет окно сообщения в сеанс, указанный в имперсонализирующем токене. Если этот флаг установлен, параметр *nWnd* должен быть равен 0. Это необходимо для того, чтобы окно сообщения могло появиться на рабочем столе, отличном от рабочего стола, соответствующего *nWnd*.

Информацию о соображениях безопасности в отношении использования этого флага см. в разделе Интерактивные службы. В частности, следует помнить, что этот флаг может создавать интерактивный контент на заблокированном рабочем столе и поэтому должен использоваться только для очень ограниченного набора сценариев, таких как исчерпание ресурсов.

сCaption (необязательно)

по умолчанию = NULL

Заголовок окна сообщения. Если *nHwnd* равен NULL, используется заголовок по умолчанию "Error".

nHwnd (необязательно)

по умолчанию = NULL

Дескриптор окна-владельца.

Если *nHwnd* равен NULL или опущен, то текущее активное окно устанавливается как окно-владелец.

Вы также можете указать 0, если диалоговое окно не должно иметь окна-владельца.

cnIconResource (необязательно)

по умолчанию = NULL

Определяет ресурс пользовательского значка.

Этот параметр может быть либо строкой, либо целочисленным идентификатором ресурса. Чтобы загрузить один из стандартных системных значков, передайте NULL в параметр *nHInstance* и укажите одно из значений, перечисленных с помощью функции [LoadIcon](#).

nHInstance (необязательно)

по умолчанию = NULL | HINSTANCE текущего исполняемого файла

Дескриптор модуля, который содержит ресурс значка, идентифицированный параметром *cnIconResource*.

Если вы опустите этот параметр или передадите NULL, но укажете пользовательский значок в параметре *cnIconResource*, *nHInstance* по умолчанию будет HINSTANCE текущего исполняемого файла.

Примечание

HINSTANCE/HMODULE модуля (dll) можно получить с помощью функции [GetModuleHandle](#) или [LoadLibrary](#).

nHelpContextId (необязательно)

по умолчанию = 0

Определяет контекст справки.

Если происходит событие справки, это значение передается окну владельца.

***nLanguageId* (необязательно)**

по умолчанию = [GetUserDefaultUILanguage](#)()

Язык, на котором будет отображаться текст, содержащийся в предопределенных кнопках.

Список поддерживаемых идентификаторов языков см. в разделе Идентификаторы языков. Обратите внимание, что каждый локализованный выпуск Windows обычно содержит ресурсы только для ограниченного набора языков. Так, например, версия для США предлагает LANG_ENGLISH, французская версия предлагает LANG_FRENCH, немецкая версия предлагает LANG_GERMAN, а японская версия предлагает LANG_JAPANESE. Каждая версия предлагает LANG_NEUTRAL. Это ограничивает набор значений, которые можно использовать с параметром *nLanguageId*. Перед указанием идентификатора языка следует перечислить локали, установленные в системе.

Возвращаемое значение

Если функция завершается успешно, возвращаемое значение представляет собой одно из следующих значений пунктов меню.

Если в окне сообщения есть кнопка «Отмена», функция возвращает значение IDCANCEL, если нажата клавиша ESC или выбрана кнопка «Отмена». Если в окне сообщения нет кнопки «Отмена», нажатие ESC не имеет никакого эффекта.

Если недостаточно памяти для создания окна сообщения, возникает ошибка 43.

Значение	Описание
IDOK	Была выбрана кнопка «ОК».
IDCANCEL	Была выбрана кнопка «Отмена».
IDABORT	Была выбрана кнопка «Отмена».
IDRETRY	Была выбрана кнопка «Повторить».
IDIGNORE	Была выбрана кнопка «Игнорировать».
IDYES	Была выбрана кнопка «Да».
IDNO	Была выбрана кнопка «Нет».
IDTRYAGAIN	Была выбрана кнопка «Попробовать еще раз».
IDCONTINUE	Была выбрана кнопка «Продолжить».

Смотрите также

Ссылки

[GetOpenFileNameEx](#)
[GetSaveFileNameEx](#)
[SHBrowseFolder](#)

Используемые функции WinApi

[MessageBoxIndirect](#)
[GetUserDefaultUILanguage](#)
[GetModuleHandle](#)

MoveFileEx

Перемещает файл, может быть передана дополнительная функция обратного вызова, которая получает статус во время выполнения операции перемещения.

```
MoveFileEx( cSourceFile , cDestinationFile [, cCallback [, nFlags  
[, nCallbackLimiter ]]])
```

Параметры

cSourceFile

Полный путь к файлу для перемещения.

cDestinationFile

Полный путь к месту назначения для операции перемещения.

cCallback (необязательно)

Функция обратного вызова с информацией о ходе выполнения во время операции перемещения.

Функция должна иметь следующий прототип:

```
FUNCTION MoveCallback( nBytesCopied, nFileSize,  
nPercentCopied)  
ENDFUNC
```

Примечание

Из функции можно вернуть следующие значения:

PROGRESS_CONTINUE — продолжает операцию перемещения

PROGRESS_CANCEL — отменяет операцию перемещения

PROGRESS_QUIET — продолжает операцию перемещения, но останавливает все дальнейшие обратные вызовы

PROGRESS_STOP — останавливает операцию перемещения

.T. — равно PROGRESS_CONTINUE

.F. — равно PROGRESS_CANCEL

nFlags (необязательно)

по умолчанию MOVEFILE_REPLACE_EXISTING | MOVEFILE_COPY_ALLOWED |
MOVEFILE_WRITE_THROUGH

Возможные значения и их смысл см. в оригинальной документации [MoveFileWithProgress](#) (параметр dwFlags).

nCallbackLimiter (необязательно)

Время в миллисекундах, которое управляет частотой вызова функции обратного вызова.

По умолчанию 100 миллисекунд.

Установка более низких значений может снизить пропускную способность копирования.

Возвращаемое значение

Статус операции перемещения.

PROGRESS_CONTINUE (0), PROGRESS_CANCEL (1), PROGRESS_STOP (2) или
PROGRESS_QUIET (3).

PROGRESS_CONTINUE и PROGRESS_QUIET означают успешное выполнение.

PROGRESS_STOP означает, что операция перемещения была остановлена, а целевой файл остался на диске.

PROGRESS_CANCEL означает, что операция перемещения была прервана, а целевой файл удален.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[SetFileAttributesEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[MoveFileWithProgress](#)

Num2Binary

Преобразует 32-битное целое число в двоичную строку, удобочитаемую человеком.

```
Num2Binary( nValue )
```

Параметры

nValue

Целое число со знаком или без знака в диапазоне от –2 147 483 648 до 4 294 967 295.

Возвращаемое значение

Строка (длиной 32 байта) с двоичным представлением переданного целочисленного значения.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

OpenRegistryKey

Открывает указанный раздел реестра.

```
OpenRegistryKey( nRegKey , cKeyName [, nAccessRights ] )
```

Параметры

nRegKey

Либо дескриптор реестра, возвращенный OpenRegistryKey, либо одна из следующих констант ключа.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и устанавливается при первой ссылке процесса на</p>

	<p>HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>
HKEY_PERFORMANCE_NLSTEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe.</p> <p>Windows 2000: этот ключ не поддерживается.</p>
HKEY_PERFORMANCE_TEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на</p>

	американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe. Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

cKeyName

Путь к подключаемому ключу ключа, переданному в *nRegKey*.

nAccessRights (необязательно)

по умолчанию = KEY_ALL_ACCESS

Одно или комбинация следующих значений.

Константа	Описание
KEY_ALL_ACCESS	Объединяет права доступа STANDARD_RIGHTS_REQUIRED, KEY_QUERY_VALUE, KEY_SET_VALUE, KEY_CREATE_SUB_KEY, KEY_ENUMERATE_SUB_KEYS, KEY_NOTIFY и KEY_CREATE_LINK.
KEY_CREATE_LINK	Зарезервировано для использования системой.
KEY_CREATE_SUB_KEY	Требуется для создания подраздела ключа реестра.
KEY_ENUMERATE_SUB_KEYS	Требуется для перечисления подразделов ключа реестра.
KEY_EXECUTE	Эквивалентно KEY_READ.
KEY_NOTIFY	Требуется для запроса уведомлений об изменениях для раздела реестра или для подразделов раздела реестра.
KEY_QUERY_VALUE	Требуется для запроса значений ключа реестра.
KEY_READ	Объединяет значения STANDARD_RIGHTS_READ, KEY_QUERY_VALUE, KEY_ENUMERATE_SUB_KEYS и KEY_NOTIFY.
KEY_SET_VALUE	Требуется для создания, удаления или установки значения реестра.
KEY_WOW64_32KEY	Указывает, что приложение в 64-разрядной версии Windows должно работать в 32-разрядном представлении реестра. Для получения дополнительной информации см. Доступ к альтернативному представлению реестра. Этот флаг должен быть объединен с помощью оператора OR с другими флагами в этой таблице, которые либо запрашивают, либо получают доступ к значениям реестра. Windows 2000: Этот флаг не поддерживается.
KEY_WOW64_64KEY	Указывает, что приложение в 64-разрядной версии Windows должно работать в 64-разрядном представлении реестра. Для получения дополнительной информации см. Доступ к альтернативному представлению реестра. Этот флаг должен быть объединен с помощью оператора OR с другими флагами в этой таблице, которые либо запрашивают, либо получают доступ к

	значениям реестра. Windows 2000: Этот флаг не поддерживается.
KEY_WRITE	Объединяет права доступа STANDARD_RIGHTS_WRITE, KEY_SET_VALUE и KEY_CREATE_SUB_KEY.

Примечание

Для получения дополнительной информации ознакомьтесь с разделом [Безопасность ключей реестра и права доступа](#).

Возвращаемое значение

Дескриптор ключа реестра.

Смотрите также**Ссылки**

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegOpenKeyEx](#)

OpenServiceEx

Открывает существующую службу.

```
OpenServiceEx( cServiceName [, nAccess [, cServer [,
cDatabase ]]])
```

Параметры

cServiceName

Название службы.

nAccess (необязательный, дополнительный)

по умолчанию = SERVICE_ALL_ACCESS

Желаемый доступ к сервису.

Одно или комбинация следующих значений.

Право доступа	Описание
SERVICE_ALL_ACCESS	Включает STANDARD_RIGHTS_REQUIRED в дополнение ко всем правам доступа в этой таблице.
SERVICE_CHANGE_CONFIG	Требуется для вызова функции ChangeServiceConfig или ChangeServiceConfig2 для изменения конфигурации службы. Поскольку это предоставляет вызывающему право изменять исполняемый файл, который запускает система, это должно быть предоставлено только администраторам.
SERVICE_ENUMERATE_DEPENDENTS	Требуется вызвать функцию ADependentServices для перечисления всех служб, зависящих от службы.
SERVICE_INTERROGATE	Требуется вызвать функцию ControlService , чтобы попросить службу немедленно сообщить о своем состоянии.
SERVICE_PAUSE_CONTINUE	Требуется для вызова функций PauseService и ContinueService .
SERVICE_QUERY_CONFIG	Требуется вызвать функцию AServiceConfig для запроса конфигурации службы.
SERVICE_QUERY_STATUS	Требуется вызвать функцию AServiceStatus , чтобы запросить у менеджера управления службами информацию о состоянии службы.
SERVICE_START	Для запуска службы необходимо вызвать функцию StartService .
SERVICE_STOP	Требуется вызвать функцию ControlService для остановки службы.
SERVICE_USER_DEFINED_CONTROL	Требуется вызвать функцию ControlService для указания определяемого пользователем кода управления.

cServer (необязательно)

Имя сервера, на котором запущена служба.

См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.
См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

Дескриптор для службы Windows.

Смотрите также

Ссылки

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[OpenService](#)
[OpenSCManager](#)

OsEx

Возвращает текущую операционную систему.

OsEx ()

Возвращаемое значение

Целое число, представляющее используемую операционную систему.

Значение	ОС
1	Windows 95
2	Windows 95 OSR2
3	Windows 98
4	Windows 98 Second Edition
5	Windows Millennium
6	Windows NT 3.5
7	Windows NT 4.0
8	Windows NT 4.0 Server
9	Windows 2000
10	Windows XP
11	Windows XP Professional x64
12	Windows Home Server
13	Windows Vista
14	Windows Server 2003
15	Windows Server 2003 R2
16	Windows 7
17	Windows Server 2008
18	Windows Server 2008 R2
19	Неизвестная Windows с более высоким номером версии

Замечания

Если действует режим совместимости, функция OsEx сообщает об операционной системе, которую она идентифицирует, что может не быть установленной операционной системой. Например, если действует режим совместимости, OsEx сообщает об операционной системе, выбранной для совместимости приложений.

Смотрите также

Ссылки

[ADesktopArea](#)
[ADesktops](#)
[ADisplayDevices](#)
[AResolutions](#)
[AWindowStations](#)
[ExpandEnvironmentStringsEx](#)
[GetLocaleInfoEx](#)
[GetSystemDirectoryEx](#)
[GetWindowsDirectoryEx](#)

Используемые функции WinApi

[GetVersionEx](#)
[GetNativeSystemInfo](#)
[GetSystemInfo](#)

PauseService

Отправляет запрос на паузу указанной службе.

```
PauseService( cServiceName | nServiceHandle [, nTimeout [, cServer  
[, cDatabase ]]])
```

Параметры

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

nTimeout (необязательно)

Максимальное время ожидания в секундах, пока служба находится в состоянии SERVICE_PAUSE_PENDING.

Если вы передадите 0, функция не будет ждать, пока служба будет приостановлена, а вместо этого вернется сразу после отправки запроса на паузу.

Если вы опустите этот параметр или передадите NULL, тайм-аут будет установлен на тайм-аут по умолчанию, сообщенный службой. См. справку MSDN для члена "dwWaitHint" структуры [SERVICE_STATUS_PROCESS](#).

cServer (необязательно)

Имя сервера, на котором запущена служба.
См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.
См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

1, если служба была приостановлена в течение периода ожидания, 0 в противном случае.

Смотрите также

Ссылки

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[StartServiceEx](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[ControlService](#)
[OpenSCManager](#)
[OpenService](#)
[CloseServiceHandle](#)

PG_ByteA2Str

Преобразует значение PostgreSQL ByteA в строку.

```
PG_ByteA2Str( cByteA )
```

Параметры

Возвращаемое значение

Строка.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

PG_Str2ByteA

Преобразует строку в экранированное значение PostgreSQL ByteA.

```
PG_Str2ByteA( cString )
```

Параметры

Возвращаемое значение

Строка.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

ProgIDFromCLSID

Извлекает ProgID для заданного CLSID.

```
ProgIDFromCLSID( cClsID | nClsIdPointer )
```

Параметры

cClsIDBinary | cClsIDString | nClsIDPointer

Функция принимает CLSID в трех различных форматах:

1. CLSID как двоичная строка (шириной 16 байт)
2. CLSID в строковом формате - {002D2B10-C1FA-4193-B134-D86EAECC5250}
3. числовой указатель, который указывает на адрес памяти CLSID в двоичной форме

Возвращаемое значение

Читаемый ProgID (строка) переданного CLSID.

Замечания

Программный идентификатор (ProgID) — это запись реестра, которая может быть связана с CLSID. Как и CLSID, ProgID идентифицирует класс, но с меньшей точностью, поскольку не гарантируется его глобальная уникальность. Формат ProgID — <Program>.<Component>.<Version>, разделенный точками и без пробелов, как в Word.Document.6.

CLSID — это глобально уникальный идентификатор, который идентифицирует объект класса COM. Если ваш сервер или контейнер позволяет ссылаться на свои встроенные объекты, вам необходимо зарегистрировать CLSID для каждого поддерживаемого класса объектов.

Пример

```
lcCLSID = CLSIDFromProgID ('VisualFoxPro.Application')
&& lcCLSID теперь содержит CLSID в двоичном формате
? ProgIDFromCLSID(lcCLSID) && преобразовать в удобочитаемый
формат
&& Возвращает VisualFoxpro.Application.9
```

Смотрите также

Ссылки

[CLSIDFromProgID](#)
[CLSIDFromString](#)
[CreateGuid](#)
[CreateThreadObject](#)
[GetIUnknown](#)
[IsEqualGuid](#)
[RegisterActiveObject](#)
[RegisterObjectAsFileMoniker](#)
[RevokeActiveObject](#)
[StringFromCLSID](#)

Используемые функции WinApi

[ProgIDFromCLSID](#)
[CoTaskMemFree](#)
[CLSIDFromString](#)

RasClearConnectionStatistics

Очищает всю накопленную статистику для указанного RAS-соединения.

```
RasClearConnectionStatistics( nRasConn )
```

Параметры

nRasConn

Действительный дескриптор соединения HRASCONN.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)

[ARasConnections](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasConnectionNotificationEx](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasGetConnectStatusEx](#)

[RasHangUpEx](#)

[RasPhonebookDlgEx](#)

Используемые функции WinApi

[RasClearConnectionStatistics](#)

RasConnectionNotificationEx

Запускает новый поток, который отслеживает наличие в системе подключений RAS. При каждом создании или завершении подключения вызывается указанная процедура обратного вызова.

```
RasConnectionNotificationEx( nRasConn , nFlags , cCallback )
```

Параметры

nRasConn

Дескриптор соединения RAS, для которого необходимо получить уведомление. Это может быть дескриптор, возвращаемый [RasDialEx](#) или [ARasConnections](#). Если этот параметр равен INVALID_HANDLE_VALUE (-1), уведомления принимаются для всех соединений RAS на локальном клиенте.

nFlags

Одно или комбинация следующих значений (определены в rasapi32.h).

Константа	Описание
RASCN_Connection	Если <i>nRasConn</i> имеет значение INVALID_HANDLE_VALUE, обратный вызов выполняется при создании любого RAS-соединения.
RASCN_Disconnection	Обратный вызов выполняется при завершении соединения <i>nRasConn</i> . Если <i>nRasConn</i> является многоканальным соединением, обратный вызов выполняется при отключении всех подзаписей. Если <i>nRasConn</i> имеет значение INVALID_HANDLE_VALUE, обратный вызов выполняется при завершении любого соединения RAS.
RASCN_BandwidthAdded	Windows NT: если <i>nRasConn</i> является дескриптором комбинированного многоканального соединения, обратный вызов выполняется при подключении подзаписи.
RASCN_BandwidthRemoved	Windows NT: если <i>nRasConn</i> является дескриптором комбинированного многоканального соединения, обратный вызов выполняется при отключении подзаписи.

cCallback

Имя функции, которая будет вызываться при создании/завершении соединения RAS. Функция должна иметь следующий прототип.

```
FUNCTION OnConnectionChange(lnConn, lnError)
    IF lnError = 0
        && вызовите ARasConnections, чтобы проверить, было
        && ли установлено новое соединение или оно было
        && отключено
        LOCAL lnConnCount, laRasConns[1,5]
        lnConnCount = ARasConnections('laRasConns')
        IF ASCAN(laRasConns,lnConn,1,0,5,8) > 0
            ? "RAS-соединение", lnConn, "подключено."
        ELSE
            ? "RAS-соединение", lnConn, "отключено."
        ENDIF
    ELSE
        ? "WaitForMultipleObjects завершился с ошибкой", ;
        lnError
```

```
ENDIF  
ENDFUNC
```

Примечание

При мониторинге одного активного RAS-подключения уведомления автоматически прекращаются после его отключения.

Возвращаемое значение

Дескриптор потока, контролирующего соединение RAS.

Смотрите также**Ссылки**

[AbortRasConnectionNotification](#)
[ARasConnections](#)
[ARasDevices](#)
[ARasPhonebookEntries](#)
[RasClearConnectionStatistics](#)
[RasDialDlgEx](#)
[RasDialEx](#)
[RasGetConnectStatusEx](#)
[RasHangUpEx](#)
[RasPhonebookDlgEx](#)

Используемые функции WinApi

[RasConnectionNotification](#)
[WaitForMultipleObjects](#)
[PostMessage](#)

RasDialDlgEx

Устанавливает RAS-соединение, используя указанную запись телефонной книги и учетные данные вошедшего в систему пользователя.

```
RasDialDlgEx([ cPhonebook [, cPhonebookEntry [, cPhoneNumber [, nSubEntry [, nHwndOwner ]]]]])
```

Параметры

cPhonebook (необязательно)

по умолчанию = пусто

Если *cPhonebook* пуст, используется системная телефонная книга по умолчанию, в противном случае следует указать допустимое имя файла телефонной книги.

Файл телефонной книги по умолчанию — это файл, выбранный пользователем на странице свойств пользовательских настроек диалогового окна Dial-Up Networking.

cPhonebookEntry (необязательно)

Имя записи телефонной книги для набора.

cPhoneNumber (необязательно)

default = пусто

Строка, которая указывает номер телефона, который переопределяет номера, сохраненные в записи телефонной книги. Если этот параметр пуст, RasDialDlgEx использует номера из записи телефонной книги.

nSubEntry (необязательно)

по умолчанию = 0

Указывает подзапись или подзаписи для набора. Если *nSubEntry* равен 0, RasDialDlgEx набирает все подзаписи, связанные с указанной записью телефонной книги.

В противном случае, чтобы указать индекс отдельной подзаписи для набора, *nSubEntry* должен быть числом от одного до количества подзаписей.

nHwndOwner (необязательно)

по умолчанию = 0

Указывает окно, которому принадлежат модальные диалоговые окна RasDialDlgEx. Этот параметр может быть любым допустимым дескриптором окна или может быть равен 0, если у диалогового окна нет владельца.

Возвращаемое значение

.T. если было установлено соединение RAS, .F. если диалог был прерван.

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)
[ARasConnections](#)
[ARasDevices](#)
[ARasPhonebookEntries](#)
[RasClearConnectionStatistics](#)
[RasConnectionNotificationEx](#)
[RasDialEx](#)
[RasGetConnectStatusEx](#)
[RasHangUpEx](#)
[RasPhonebookDlgEx](#)

Используемые функции WinApi[RasDialDlg](#)

RasDialEx

Функция [RasDial](#) устанавливает RAS-соединение между RAS-клиентом и RAS-сервером.

```
RasDialEx( cPhonebookEntry | nRasDialParamsPtr [, cPhonebook [,
cCallback [, nFlags [, nRasConn ]]])
```

Параметры

cPhonebookEntry | nRasDialParamsPtr

Либо имя записи телефонной книги, либо указатель на структуру [RASDIALPARAMS](#).

cPhonebook (необязательно)

по умолчанию = пусто

Если *cPhonebook* пуст, используется системная телефонная книга по умолчанию, в противном случае следует указать допустимое имя файла телефонной книги. Файл телефонной книги по умолчанию — это файл, выбранный пользователем на странице свойств пользовательских настроек диалогового окна Dial-Up Networking.

cCallback (необязательно)

по умолчанию = во время операции набора номера обратные вызовы не выполняются.

Функция FoxPro для обратного вызова.

Если вы передаете функцию обратного вызова, функция [RasDial](#) выполняется асинхронно.

Она немедленно вернется, и во время процесса набора номера будет вызвана ваша функция обратного вызова, которая предоставит вам текущий статус операции.

См. класс RasConnection:DialCallback в примере ras.prg.

nFlags (необязательно)

по умолчанию = 0

Один или комбинация следующих флагов.

Константа	Описание
RDEOPT_UsePrefixSuffix	Если этот флаг установлен, RasDialEx использует префикс и суффикс, которые есть в телефонной книге RAS. Если этот флаг не установлен, RasDialEx игнорирует префикс и суффикс, которые есть в телефонной книге RAS. Если в вызове RasDialEx не указано имя записи телефонной книги, фактическое значение этого флага игнорируется и предполагается равным нулю.
RDEOPT_PausedStates	Если этот флаг установлен, RasDialEx принимает приостановленные состояния. Примерами приостановленных состояний являются режим терминала, повторный вход в систему, смена пароля, установка номера обратного вызова и аутентификация EAP. Если этот флаг не установлен, RasDialEx сообщает о фатальной ошибке, если он входит в приостановленное состояние.
RDEOPT_IgnoreModemSpeaker	Если этот флаг установлен, RasDialEx игнорирует настройку динамика модема, которая находится в телефонной книге RAS, и использует настройку, указанную битовым флагом

	<p>RDEOPT_SetModemSpeaker.</p> <p>Если этот флаг не установлен, RasDialEx использует настройку динамика модема, которая находится в телефонной книге RAS, и игнорирует настройку, указанную битовым флагом RDEOPT_SetModemSpeaker. Если в вызове RasDialEx не указано имя записи телефонной книги, выбор делается между использованием настройки по умолчанию или настройки, указанной битовым флагом RDEOPT_SetModemSpeaker. Настройка по умолчанию используется, если RDEOPT_IgnoreModemSpeaker равен нулю. Настройка, указанная RDEOPT_SetModemSpeaker, используется, если RDEOPT_IgnoreModemSpeaker равен единице.</p>
RDEOPT_SetModemSpeaker	<p>Если этот флаг установлен, а RDEOPT_IgnoreModemSpeaker равен единице, RasDialEx включает динамик модема.</p> <p>Если этот флаг не установлен, а RDEOPT_IgnoreModemSpeaker равен единице, RasDialEx выключает динамик модема. Если RDEOPT_IgnoreModemSpeaker не установлен, RasDialEx игнорирует значение RDEOPT_SetModemSpeaker и устанавливает динамик модема на основе настройки телефонной книги RAS или настройки по умолчанию.</p>
RDEOPT_IgnoreSoftwareCompression	<p>Если этот флаг установлен, RasDialEx игнорирует настройку программного сжатия, которая находится в телефонной книге RAS, и использует настройку, указанную битовым флагом RDEOPT_SetSoftwareCompression.</p> <p>Если этот флаг не установлен, RasDialEx использует настройку программного сжатия, которая находится в телефонной книге RAS, и игнорирует настройку, указанную флагом RDEOPT_SetSoftwareCompression.</p> <p>Если в вызове RasDialEx не указано имя записи телефонной книги, выбор делается между использованием настройки по умолчанию или настройки, указанной битовым флагом RDEOPT_SetSoftwareCompression. Настройка по умолчанию используется, если RDEOPT_IgnoreSoftwareCompression равен нулю. Настройка, указанная RDEOPT_SetSoftwareCompression, используется, если RDEOPT_IgnoreSoftwareCompression равен единице.</p>
RDEOPT_SetSoftwareCompression	<p>Если этот флаг установлен, а RDEOPT_IgnoreSoftwareCompression равен единице, RasDialEx использует программное сжатие. Если этот флаг не установлен, а RDEOPT_IgnoreSoftwareCompression равен единице, RasDialEx не использует программное сжатие.</p> <p>Если RDEOPT_IgnoreSoftwareCompression равен нулю, RasDialEx игнорирует значение RDEOPT_SetSoftwareCompression и устанавливает</p>

	состояние программного сжатия на основе настройки телефонной книги RAS или настройки по умолчанию.
RDEOPT_UseCustomScripting	<p>Если этот флаг установлен, RasDialEx вызывает DLL-библиотеку пользовательских сценариев после установления соединения с сервером. RasDialEx вызывает DLL только при соблюдении всех следующих условий:</p> <p>Этот флаг установлен. DLL-библиотека пользовательских сценариев правильно зарегистрирована в системе. Параметр RASEO_CustomScript указан в записи телефонной книги.</p>
RDEOPT_CustomDial	Если этот флаг установлен, RasDialEx набирает номер обычным образом, а не вызывает точку входа RasCustomDial пользовательского номеронабирателя. Этот флаг используется, когда RasCustomDialDlg вызывает RasDialEx, чтобы RasDialEx не приходилось вызывать пользовательский номеронабиратель обратно.
RDEOPT_NoUser	Windows Vista или более поздняя версия: Если этот флаг установлен, соединение было установлено из контекста Windows Logon. В этом случае RasDialEx использует пользовательские настройки по умолчанию для соединения. Этот флаг должен быть установлен, если RCD_Logon установлен в параметре dwfOptions RasCustomDialDlg.

***nRasConn* (необязательно)**

по умолчанию = не используется.

Передайте существующий дескриптор HRASCONN при возобновлении асинхронного вызова RasDialEx из приостановленного состояния.

Возвращаемое значение

Дескриптор (HRASCONN), представляющий соединение RAS.

Пример

Набор индивидуального номера.

```
LOCAL lnConn, loRasParams
loRasParams = CREATEOBJECT ('RASDIALPARAMS')
loRasParams.szPhoneNumber = '111 2453'
loRasParams.szCallbackNumber = '*'
loRasParams.szUserName = 'имя пользователя'
loRasParams.szPassword = '*****'
loRasParams.szDomain = ''
lnConn = RasDialEx(loRasParams.Address)
```

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)
[ARasConnections](#)
[ARasDevices](#)
[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)
[RasConnectionNotificationEx](#)
[RasDialDlgEx](#)
[RasGetConnectStatusEx](#)
[RasHangUpEx](#)
[RasPhonebookDlgEx](#)

Используемые функции WinApi

[RasDial](#)
[RasGetEntryDialParams](#)
[RasGetEapUserIdentity](#)
[RasGetConnectStatus](#)
[RasHangUp](#)
[RasFreeEapUserIdentity](#)

RasGetConnectStatusEx

Извлекает информацию о текущем состоянии указанного подключения удаленного доступа в массив.

```
RasGetConnectStatusEx( nRasConn , cArrayname )
```

Параметры

nRasConn

Действительный дескриптор HRASCONN, для которого необходимо получить информацию о состоянии.

cArrayname

По возвращении массив содержит следующую информацию из структуры [RASCONNSTATUS](#).

Элемент	Значение
1	RASCONNSTATE
2	Код ошибки
3	Тип устройства
4	Имя устройства
5	Номер телефона

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)
[ARasConnections](#)
[ARasDevices](#)
[ARasPhonebookEntries](#)
[RasClearConnectionStatistics](#)
[RasConnectionNotificationEx](#)
[RasDialDlgEx](#)
[RasDialEx](#)
[RasHangUpEx](#)
[RasPhonebookDlgEx](#)

Используемые функции WinApi

[RasGetConnectStatus](#)

RasHangUpEx

Завершает соединение удаленного доступа.

```
RasHangUpEx ( nRasConn )
```

Параметры

nRasConn

Действительный дескриптор соединения HRASCONN.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)

[ARasConnections](#)

[ARasDevices](#)

[ARasPhonebookEntries](#)

[RasClearConnectionStatistics](#)

[RasConnectionNotificationEx](#)

[RasDialDlgEx](#)

[RasDialEx](#)

[RasGetConnectStatusEx](#)

[RasPhonebookDlgEx](#)

Используемые функции WinApi

[RasHangUp](#)

[RasGetConnectStatus](#)

RasPhonebookDlgEx

Отображает главное диалоговое окно Dial-Up Networking.

Из этого модального диалогового окна пользователь может набрать номер, изменить или удалить выбранную запись телефонной книги, создать новую запись телефонной книги или указать пользовательские настройки. Функция возвращается, когда диалоговое окно закрывается.

```
RasPhonebookDlgEx([ cInitialSelectedEntry [, cPhonebook [,  
cCallback [, nFlags ]]])
```

Параметры

***cInitialSelectedEntry* (необязательно)**

по умолчанию = пусто

Имя записи телефонной книги, которая должна быть изначально выбрана в диалоговом окне.

***cPhonebook* (необязательно)**

по умолчанию = пусто

Если *cPhonebook* пуст, используется системная телефонная книга по умолчанию, в противном случае следует указать допустимое имя файла телефонной книги. Файл телефонной книги по умолчанию — это файл, выбранный пользователем на странице свойств пользовательских настроек диалогового окна Dial-Up Networking.

***cCallback* (необязательно)**

по умолчанию = пусто.

Если этот параметр пуст, обратные вызовы из диалога не производятся. В противном случае переданная функция вызывается обратно во время работы диалога. Функция должна иметь следующий прототип.

```
FUNCTION DialogCallback(lnEvent, lcText, lnData)
  DO CASE
    CASE lnEvent = RASPBDEVENT_AddEntry
      ? "RasEntry добавлен: ", lcText
    CASE lnEvent = RASPBDEVENT_EditEntry
      ? "RasEntry отредактировано: ", lcText
    CASE lnEvent = RASPBDEVENT_RemoveEntry
      ? "RasEntry удален: ", lcText
    CASE lnEvent = RASPBDEVENT_DialEntry
      ? "RasEntry набран: ", lcText
    CASE lnEvent = RASPBDEVENT_EditGlobals
      ? "RasEntry (глобальный) отредактировано:", lcText
    CASE lnEvent = RASPBDEVENT_NoUser
      ? "Событие RasEntry NoUser"
    CASE lnEvent = RASPBDEVENT_NoUserEdit
      ? "Событие RasEntry NoUserEdit "
  ENDCASE
ENDFUNC
```

***nFlags* (необязательно)**

по умолчанию = 0

Одно или комбинация следующих значений (определены в rasapi32.h).

Константа	Описание
-----------	----------

RASPBDFLAG_PositionDlg	Заставляет RasPhonebookDlg использовать значения, указанные членами xDlg и yDlg, для позиционирования диалогового окна. Если этот флаг не установлен, диалоговое окно центрируется на окне владельца, если только hwndOwner не равен NULL, в этом случае диалоговое окно центрируется на экране.
RASPBDFLAG_ForceCloseOnDial	Включает опцию close-on-dial, переопределяя предпочтения пользователя. Эта опция подходит для таких функций, как RAS AutoDial, где цель пользователя — немедленно установить соединение.
RASPBDFLAG_NoUser	Заставляет функцию обратного вызова RasPBDlgFunc, указанную членом pCallback, получать уведомление RASPBDEVENT_NoUser при запуске диалогового окна. Этот флаг используется в ситуациях, когда нет вошедшего в систему пользователя, как в приложении WinLogon. Обычно приложения не должны использовать этот флаг.
RASPBDFLAG_UpdateDefaults	Заставляет сохранять позицию окна по умолчанию при выходе. Этот флаг используется в основном RASPHONE.EXE и не должен использоваться типичными приложениями.

Возвращаемое значение

.T. если было установлено соединение RAS, .F. если диалог был прерван.

Смотрите также

Ссылки

[AbortRasConnectionNotification](#)
[ARasConnections](#)
[ARasDevices](#)
[ARasPhonebookEntries](#)
[RasClearConnectionStatistics](#)
[RasConnectionNotificationEx](#)
[RasDialDlgEx](#)
[RasDialEx](#)
[RasGetConnectStatusEx](#)
[RasHangUpEx](#)

Используемые функции WinApi

[RasPhonebookDlg](#)

ReadBytes

Возвращает диапазон байтов по указанному адресу.

```
ReadBytes( nAddress , nLen )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nLen

Количество байтов для чтения

Возвращаемое значение

Строка.

Смотрите также

Ссылки

[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)

[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadChar

Возвращает символ С (отдельный символ) из указанного адреса.

```
ReadChar ( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Строка.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadCharArray

Извлекает строку из массива символов в стиле C.

```
ReadCharArray( nAddress , nLen )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nLen

Максимальная длина массива символов.

Функция считывает до *nLen* символов или меньше, если найден завершающий символ 0.

Возвращаемое значение

Строка.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadCString

Возвращает строку С из указанного адреса.

```
ReadCString( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Строка.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadDouble

Возвращает число двойной точности (64-битное значение с плавающей точкой) из указанного адреса.

```
ReadDouble( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Число.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)

[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadFloat

Возвращает число с плавающей точкой (32-битное значение с плавающей точкой) из указанного адреса.

```
ReadFloat( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Число.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)

[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadInt

Извлекает 32-битное целое число из указанного адреса.

```
ReadInt ( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Число.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadInt64

Извлекает 64-битное целое число со знаком из указанного адреса.

```
ReadInt64( nAddress [, nFormat ])
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	8-байтовая двоичная строка.
3	Строковый литерал.
4	Двойной.

Примечание

Если вы передадите 4, значение может быть усечено!

Возвращаемое значение

Значение считывается в запрошенном формате.

Замечания

Возвращаемое функцией значение валюты должно использоваться только со следующими функциями:

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int642Str](#), [WriteInt64](#), [WritePInt64](#) и [WriteRegistryKey](#).

Тип данных валюты VFP имеет подразумеваемую десятичную точку. При преобразовании значения валюты, возвращаемого из ReadInt64 или [ReadPInt64](#) с помощью [MTON](#)(), возвращаемое числовое значение не соответствует ожидаемому, кроме того, каждая функция VFP выдаст ошибку 1988, если значение равно -9 223 372 036 854 775 808.

Если вы планируете хранить 64-битные целые числа в таблице, используйте тип данных Q(8) вместо Y.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)

[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadInt8

Извлекает 8-битное целое число из указанного адреса.

```
ReadInt8( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Число.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadLogical

Извлекает логическое значение из указанного адреса.

```
ReadLogical( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Логическое.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPChar

Возвращает символ С (отдельный символ) из указанного косвенного адреса.

```
ReadPChar( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Строка или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPCString

Возвращает строку С из указанного косвенного адреса.

```
ReadPCString( nAddress [, vDefault ] )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

***vDefault* (необязательно)**

по умолчанию = пустая строка

Значение, которое нужно вернуть, если указатель на *nAddress* равен 0.

Возвращаемое значение

Строка или *vDefault* .

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)

[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPDouble

Возвращает число двойной точности (64-битное значение с плавающей точкой) из указанного косвенного адреса.

```
ReadPDouble( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)

[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPFloat

Возвращает число с плавающей точкой (32-битное значение с плавающей точкой) из указанного косвенного адреса.

```
ReadPFloat( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)

[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPInt

Извлекает 32-битное целое число из указанного косвенного адреса.

```
ReadPInt ( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPInt64

Извлекает 64-битное целое число со знаком из указанного косвенного адреса.

```
ReadPInt64( nAddress [, nFormat ])
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	8-байтовая двоичная строка.
3	Строковый литерал.
4	Двойной.

Примечание

Если вы передадите 4, значение может быть усечено!

Возвращаемое значение

Значение считывается в запрошенном формате или NULL.

Замечания

Возвращаемое функцией значение валюты должно использоваться только со следующими функциями:

[Int64_Add](#), [Int64_Sub](#), [Int64_Mul](#), [Int64_Div](#), [Int64_Mod](#), [Int642Str](#), [WriteInt64](#), [WritePInt64](#) и [WriteRegistryKey](#).

Тип данных валюты VFP имеет подразумеваемую десятичную точку. При преобразовании значения валюты, возвращаемого из [ReadInt64](#) или [ReadPInt64](#) с помощью [MTON\(\)](#), возвращаемое числовое значение не соответствует ожидаемому, кроме того, каждая функция VFP выдаст ошибку 1988, если значение равно -9 223 372 036 854 775 808.

Если вы планируете хранить 64-битные целые числа в таблице, используйте тип данных Q(8) вместо Y.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)

[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUSHort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPInt8

Извлекает 8-битное целое число из указанного косвенного адреса.

```
ReadPInt8( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPLogical

Извлекает логическое значение из указанного косвенного адреса.

```
ReadPLogical( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Логический или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPointer

Извлекает указатель из указанного адреса.

```
ReadPointer( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPPointer

Извлекает указатель из указанного косвенного адреса.

```
ReadPPointer( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadProcessMemoryEx

Извлекает диапазон байтов из области памяти другого процесса.

```
ReadProcessMemoryEx( nProcessID , nAddress , nNoOfBytes )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nNoOfBytes

Количество байтов для чтения.

Возвращаемое значение

Строка.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)

[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[Toolhelp32ReadProcessMemory](#)

ReadPShort

Извлекает 16-битное целое число из указанного косвенного адреса.

```
ReadPShort ( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPUInt

Извлекает 32-битное беззнаковое целое число из указанного косвенного адреса.

```
ReadPUInt( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPUInt64

Извлекает 64-битное беззнаковое целое число из указанного косвенного адреса.

```
ReadPUInt64( nAddress [, nFormat ])
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	8-байтовая двоичная строка.
3	Строковый литерал.
4	Двойной.

Примечание

Если вы передадите 4, значение может быть усечено!

Возвращаемое значение

Значение считывается в запрошенном формате или NULL.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt8](#)
[ReadPUShort](#)

[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPUInt8

Извлекает 8-битное беззнаковое целое число из указанного косвенного адреса.

```
ReadPUInt8 ( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPUSHort

Извлекает 16-битное целое число без знака из указанного косвенного адреса.

```
ReadPUSHort ( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое или NULL.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadPWString

Извлекает строку Unicode, преобразованную в Ansi, из указанного косвенного адреса.

```
ReadPWString( nAddress [, nCodePage [, vDefault ]])
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

***nCodePage* (необязательно)**

по умолчанию = кодовая страница, заданная с помощью [VFP2CSys](#)(4, *nCodePage*),
которая по умолчанию равна CP_ACP

Кодовая страница, используемая при выполнении преобразования Unicode в ANSI. Этот параметр может быть установлен на значение любой кодовой страницы, установленной или доступной в операционной системе.

***vDefault* (необязательно)**

по умолчанию = пустая строка

Значение, которое нужно вернуть, если указатель на *nAddress* равен 0.

Возвращаемое значение

Строка или *vDefault* .

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)

[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[WideCharToMultiByte](#)

ReadRegistryKey

Извлекает данные для указанного значения реестра.

```
ReadRegistryKey( nRegKey [, cValueName [, cKeyName [,
cValueType ]]])
```

Параметры

nRegKey

Либо дескриптор реестра, возвращенный [OpenRegistryKey](#), либо одна из следующих констант ключа.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и</p>

	<p>устанавливается при первой ссылке процесса на HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>
HKEY_PERFORMANCE_NLSTEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe.</p> <p>Windows 2000: этот ключ не поддерживается.</p>
HKEY_PERFORMANCE_TEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются</p>

	на текстовые строки, описывающие счетчики на американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe.
	Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

cValueName

Имя значения, для которого возвращаются данные.

Если вы передаете пустую строку, возвращается значение по умолчанию указанного ключа.

cKeyName

Путь к подключаемому ключу ключа, переданному в *nRegKey*.

cValueType (необязательно)

Тип данных VFP, в который должно быть преобразовано значение реестра.

Если вы опустите *cValueType*, он будет извлечен из реестра.

В следующей таблице перечислены преобразования реестра в типы VFP по умолчанию.

Тип реестра	Тип VFP
REG_SZ, REG_MULTI_SZ, REG_EXPAND_SZ, REG_LINK и REG_NONE	C
REG_BINARY	Q **
REG_DWORD, REG_DWORD_BIG_ENDIAN, REG_DOUBLE	N
REG_INTEGER	I
REG_LOGICAL	L
REG_DATE	D
REG_DATETIME	T
REG_QWORD, REG_MONEY	Y

Если вы передадите *cValueType*, в следующей таблице будут перечислены допустимые преобразования.

Тип реестра	cValueType
REG_SZ	C, Q
REG_EXPAND_SZ	C, Q
REG_MULTI_SZ	C, Q
REG_BINARY	C, Q
REG_DWORD	N, I, L, Q
REG_DWORD_BIG_ENDIAN	N, I, L, Q
REG_QWORD	Y, Q, C, N
REG_INTEGER	I, N, L, Q
REG_LOGICAL	I, N, L, Q
REG_DOUBLE	N, D, T, Q

REG_DATE	D, T, Q
REG_DATETIME	D, T, Q
REG_MONEY	Y, Q

Примечание

** Поскольку невозможно вернуть значения varbinary (Q) из FLL, значение возвращается как обычная строка.
Вы можете преобразовать его в значение varbinary с помощью [CREATEBINARY](#) ().

Возвращаемое значение

Данные для значения реестра.

Смотрите также

Ссылки

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegOpenKeyEx](#)
[RegQueryValueEx](#)
[RegCloseKey](#)

ReadShort

Извлекает 16-битное целое число из указанного адреса.

```
ReadShort( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadUInt

Извлекает 32-битное беззнаковое целое число из указанного адреса.

```
ReadUInt ( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadUInt64

Извлекает 64-битное беззнаковое целое число из указанного адреса.

```
ReadUInt64( nAddress [, nFormat ])
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	8-байтовая двоичная строка.
3	Строковый литерал.
4	Двойной.

Примечание

Если вы передадите 4, значение может быть усечено!

Возвращаемое значение

Значение считывается в запрошенном формате.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)

[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadUInt8

Извлекает 8-битное беззнаковое целое число из указанного адреса.

```
ReadUInt8( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)
- [WriteInt64](#)
- [WriteInt8](#)
- [WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadUShort

Извлекает 16-битное целое число без знака из указанного адреса.

```
ReadUShort ( nAddress )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

Возвращаемое значение

Числовое.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)

[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

ReadWCharArray

Извлекает строку из массива символов Unicode в стиле C.

```
ReadWCharArray( nAddress , nLen [, nCodePage ] )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nLen

Максимальная длина массива символов.

Функция считывает до *nLen* символов или меньше, если найден завершающий символ 0.

***nCodePage* (необязательно)**

по умолчанию = кодовая страница, заданная с помощью [VFP2CSys](#)(4, *nCodePage*), которая по умолчанию равна CP_ACP

Кодовая страница, используемая при выполнении преобразования Unicode в ANSI. Этот параметр может быть установлен на значение любой кодовой страницы, установленной или доступной в операционной системе.

Возвращаемое значение

Строка.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)

[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[WideCharToMultiByte](#)

ReadWString

Извлекает строку Unicode, преобразованную в Ansi, из указанного адреса.

```
ReadWString( nAddress [, nCodePage ] )
```

Параметры

nAddress

Адрес памяти, из которого следует прочитать значение.

nCodePage (необязательно)

по умолчанию = кодовая страница, заданная с помощью [VFP2CSys](#)(4, nCodePage),
которая по умолчанию равна CP_ACP

Кодовая страница, используемая при выполнении преобразования Unicode в ANSI. Этот параметр может быть установлен на значение любой кодовой страницы, установленной или доступной в операционной системе.

Возвращаемое значение

Строка.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)

[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[WideCharToMultiByte](#)

ReAllocHGlobal

Изменяет размер указанного объекта глобальной памяти.

```
ReAllocHGlobal( nHandle , nNumberOfBytes [, nFlags ])
```

Параметры

nHandle

Дескриптор блока памяти, возвращаемый [AllocHGlobal](#).

nNumberOfBytes

Новый размер блока памяти.

nFlags (необязательно)

по умолчанию = GMEM_ZEROINIT

Параметры перераспределения.

Если указано GMEM_MODIFY, функция изменяет только атрибуты объекта памяти (параметр *nNumberOfBytes* игнорируется.) В противном случае функция перераспределяет объект памяти.

При желании можно объединить GMEM_MODIFY со следующим значением.

Флаг	Описание
GMEM_MOVEABLE	Выделяет перемещаемую память. Если память является заблокированным блоком памяти GMEM_MOVEABLE или блоком памяти GMEM_FIXED и этот флаг не указан, память может быть перераспределена только на месте.

Если этот параметр не указывает GMEM_MODIFY, можно использовать следующее значение.

Флаг	Описание
GMEM_ZEROINIT	Приводит к инициализации дополнительного содержимого памяти нулем, если размер объекта памяти увеличивается.

Возвращаемое значение

Дескриптор или указатель на выделенную область памяти.

Замечания

Если ReAllocHGlobal перераспределяет перемещаемый объект, возвращаемое значение — дескриптор объекта памяти. Чтобы преобразовать дескриптор в указатель, используйте функцию [LockHGlobal](#).

Если ReAllocHGlobal перераспределяет фиксированный объект, возвращаемое значение дескриптора — адрес первого байта блока памяти.

Если ReAllocHGlobal терпит неудачу, исходная память не освобождается, а исходные дескриптор и указатель остаются действительными.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[GlobalReAlloc](#)

ReAllocMem

Изменяет размер блока памяти, ранее выделенного [AllocMem](#).

```
ReAllocMem( nAddress , nNumberOfBytes )
```

Параметры

nAddress

Указатель на блок памяти, возвращаемый [AllocMem](#).

nNumberOfBytes

Новый размер блока памяти.

Возвращаемое значение

Указатель (числовой) на запрошенный блок памяти.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[SizeOfMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapReAlloc](#)

RegisterActiveObject

Регистрирует переданный объект как активный объект для класса, указанного в cProgID. Регистрация приводит к тому, что объект указывается в таблице запущенных объектов (ROT) OLE, глобально доступной таблице поиска, которая отслеживает объекты, которые в данный момент запущены на компьютере.

```
RegisterActiveObject( oObjectReference , cProgID )
```

Параметры

oObjectReference

Ссылка на COM-объект, который вы хотите зарегистрировать в ROT (таблица запущенных объектов).

cProgID

ProgID вашего COM-объекта — та же строка, которую вы использовали в [CREATEOBJECT](#).

Возвращаемое значение

Числовой идентификатор, представляющий зарегистрированный объект.

Пример

```
LOCAL yourObject, nObjectHandle
yourObject = CREATEOBJECT('some.ComObject')
nObjectHandle = RegisterActiveObject(m.yourObject, ;
    'some.ComObject')
```

&& объект теперь можно получить из любой программы с помощью

```
LOCAL loComObjectRef
m.loComObjectRef = GETOBJECT(, 'some.ComObject')
```

Смотрите также

Ссылки

[CLSIDFromProgID](#)
[CLSIDFromString](#)
[CreateGuid](#)
[CreateThreadObject](#)
[GetIUnknown](#)
[IsEqualGuid](#)
[ProgIDFromCLSID](#)
[RegisterObjectAsFileMoniker](#)
[RevokeActiveObject](#)
[StringFromCLSID](#)

Используемые функции WinApi

[RegisterActiveObject](#)
[CLSIDFromProgID](#)

RegisterObjectAsFileMoniker

Создает файловое имя для переданного объекта.

```
RegisterObjectAsFileMoniker( oObjectReference , cProgID ,  
cFileName )
```

Параметры

cObjectReference

Ссылка на COM-объект, который вы хотите зарегистрировать как файловое имя.

cProgID

ProgID вашего COM-объекта — та же строка, которую вы использовали в [CREATEOBJECT](#).

cFileName

Имя файла.

Возвращаемое значение

Числовой идентификатор, представляющий зарегистрированный объект.

Замечания

Файловые моникеры могут использоваться для идентификации любого объекта, который хранится в собственном файле. Файловый моникер действует как оболочка для имени пути, которое собственная файловая система назначает файлу. Источник объекта, названного моникером, должен предоставлять реализацию интерфейса IPersistFile для поддержки привязки файлового моникера. Файловые моникеры могут представлять как полный, так и относительный путь.

Пример

```
LOCAL yourObject, nObjectHandle  
yourObject = CREATEOBJECT('some.ComObject')  
nObjectHandle= RegisterObjectAsFileMoniker(m.yourObject, ;  
    'some.ComObject', 'D:\objec.ref')  
? nObjectHandle  
  
&& объект теперь можно получить из любой программы с помощью  
LOCAL loComObjectRef  
m.loComObjectRef = GETOBJECT('D:\object.ref')
```

Смотрите также

Ссылки

[CLSIDFromProgID](#)
[CLSIDFromString](#)
[CreateGuid](#)
[CreateThreadObject](#)
[GetIUnknown](#)
[IsEqualGuid](#)
[ProgIDFromCLSID](#)
[RegisterActiveObject](#)
[RevokeActiveObject](#)
[StringFromCLSID](#)

Используемые функции WinApi

[CLSIDFromProgID](#)
[GetRunningObjectTable](#)

[StgOpenStorage](#)
[StgCreateDocfile](#)
[WriteClassStg](#)
[ReadClassStg](#)
[CreateFileMoniker](#)
[IRunningObjectTable](#) Register

RegistryHiveToObject

Сохраняет раздел реестра, включая все подразделы, в объекте.

```
RegistryHiveToObject( nRegKey , cKeyName , oObject )
```

Параметры

nRegKey

Либо дескриптор реестра, возвращенный [OpenRegistryKey](#), либо одна из следующих констант ключа.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и устанавливается при первой ссылке процесса на</p>

	<p>HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>
HKEY_PERFORMANCE_NLSTEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe.</p> <p>Windows 2000: этот ключ не поддерживается.</p>
HKEY_PERFORMANCE_TEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на</p>

	американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe.
	Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

cKeyName

Путь к подключаемому ключу ключа, переданному в *nRegKey* .

oObject

Объект, который расширяется свойствами, соответствующими прочитанным значениям реестра.

Примечание

Так как имена ключей и значений реестра могут содержать символы, недопустимые для имени свойства VFP, функция преобразует имена.

Схема преобразования следующая:

1. Если имя значения начинается с цифры, добавляется подчеркивание (_)
2. Каждый недопустимый символ (за исключением 0-9, az, AZ и _) заменяется подчеркиванием, например, "2.SomeName" -> "_2_SomeName", "{HelloWorld}" -> "_HelloWorld_"

Возвращаемое значение

Всегда .T.

Смотрите также**Ссылки**

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryValuesToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegOpenKeyEx](#)
[RegQueryInfoKey](#)
[RegEnumValue](#)
[RegEnumKeyEx](#)
[RegCloseKey](#)

RegistryValuesToObject

Сохраняет все значения ключа реестра в объекте.

```
RegistryValuesToObject( nRegKey , cKeyName , oObject )
```

Параметры

nRegKey

Либо дескриптор реестра, возвращенный [OpenRegistryKey](#), либо одна из следующих констант ключа.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и устанавливается при первой ссылке процесса на</p>

	<p>HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>
HKEY_PERFORMANCE_NLSTEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe.</p> <p>Windows 2000: этот ключ не поддерживается.</p>
HKEY_PERFORMANCE_TEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на</p>

	американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe.
	Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

cKeyName

Путь к подключаемому ключу ключа, переданному в *nRegKey*.

oObject

Объект, который расширяется свойствами, соответствующими прочитанным значениям реестра.

Примечание

Так как имена ключей и значений реестра могут содержать символы, недопустимые для имени свойства VFP, функция преобразует имена.

Схема преобразования следующая:

1. Если имя значения начинается с цифры, добавляется подчеркивание (_)
2. Каждый недопустимый символ (за исключением 0-9, az, AZ и _) заменяется подчеркиванием, например, "2.SomeName" -> "_2_SomeName", "{HelloWorld}" -> "_HelloWorld_"

Возвращаемое значение

Количество значений реестра, добавленных к объекту.

Смотрите также**Ссылки**

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[WriteRegistryKey](#)

Используемые функции WinApi

[RegOpenKeyEx](#)
[RegQueryInfoKey](#)
[RegEnumValue](#)
[RegCloseKey](#)

ReleasePublicShadowObjReference

Освобождает публичную переменную, ссылающуюся на объект, созданный с помощью [CreatePublicShadowObjReference](#).

```
ReleasePublicShadowObjReference ( cVariableName )
```

Параметры

cVariableName

Имя переменной, которую необходимо освободить.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AsyncWaitForObject](#)
[BindEventsEx](#)
[CancelWaitForObject](#)
[CreateCallbackFunc](#)
[CreatePublicShadowObjReference](#)
[DestroyCallbackFunc](#)
[UnbindEventsEx](#)

ResolveHostToIp

Определяет IP-адрес для указанного имени хоста.

```
ResolveHostToIp( cHostname [, cArrayName ] )
```

Параметры

cHostname

Имя хоста, например «www.google.com» или сервера в вашей сети

cArrayname (необязательно)

Если передать имя массива, функция вернет количество IP-адресов и сохранит IP-адреса в массиве.

Возвращаемое значение

Если имя массива не было передано в качестве второго параметра, то IP (строка) или пустая строка, если хост не может быть разрешен.

Если имя массива было передано, то количество IP-адресов для хоста или 0, если хост не может быть разрешен.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)

[AllIpAddresses](#)

[AllNetFiles](#)

[AllNetServers](#)

[GetServerTime](#)

[ICmpPing](#)

[Ip2MacAddress](#)

[SyncToSNTPServer](#)

[UrlDownloadToFileEx](#)

Используемые функции WinApi

[gethostbyname](#)

[inet_ntoa](#)

RevokeActiveObject

Отменяет регистрацию объекта в таблице текущих объектов (ROT).

```
RevokeActiveObject( hObject )
```

Параметры

nObjectHandle

Числовое значение, возвращаемое [RegisterActiveObject](#).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[CLSIDFromProgID](#)

[CLSIDFromString](#)

[CreateGuid](#)

[CreateThreadObject](#)

[GetIUnknown](#)

[IsEqualGuid](#)

[ProgIDFromCLSID](#)

[RegisterActiveObject](#)

[RegisterObjectAsFileMoniker](#)

[StringFromCLSID](#)

Используемые функции WinApi

[RevokeActiveObject](#)

RGB2Colors

Разбивает RGB на составные части.

```
RGB2Colors( nRGB, @Red, @Green, @Blue [, @Alpha ] )
```

Параметры

nRGB

Комбинированное значение цвета.

@Red

Переменная по ссылке, в которой хранится красная часть цвета.

@Green

Переменная по ссылке, в которой хранится зеленая часть цвета.

@Blue

Переменная по ссылке, в которой хранится синяя часть цвета.

@Alpha (необязательно)

Переменная по ссылке, в которой хранится альфа-канал цвета.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

SetFileAttributesEx

Изменяет атрибуты указанного файла или набора файлов/каталогов, указанных с помощью подстановочного знака поиска.

```
SetFileAttributesEx( cFileName | cFileSkeleton , ncAttributes [,
nAttributesRemove [, nMaxRecursion ]])
```

Параметры

cFileName | cFileSkeleton

Имя файла или подстановочный знак, для которого необходимо задать атрибуты.

Примечание

Это может быть относительный путь или только имя файла, функция VFP [FULLPATH](#) используется для получения полного имени файла. Используется обычный путь поиска VFP ([SET PATH](#)).

Полная строка поиска (диск:\путь\подстановочный знак), например "C:\Winnt*.dll"

ncAttributes (дополнительные)

Одна или комбинация следующих констант или строка, где каждый символ представляет атрибут.

При передаче строки знак '-' перед атрибутом удаляет атрибут из текущих атрибутов файлов, а знак '+' добавляет атрибут. При передаче числового значения атрибуты файла перезаписываются новым значением

Атрибут	Символ	Описание
FILE_ATTRIBUTE_ARCHIVE	A	Файл или каталог, который является архивным файлом или каталогом. Приложения обычно используют этот атрибут, чтобы пометить файлы для резервного копирования или удаления.
FILE_ATTRIBUTE_HIDDEN	H	Файл или каталог скрыт. Он не включен в обычный список каталогов.
FILE_ATTRIBUTE_NORMAL	N	Файл, у которого не установлены другие атрибуты. Этот атрибут действителен только при использовании отдельно.
FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	I	Файл или каталог не подлежит индексации службой индексации контента.
FILE_ATTRIBUTE_OFFLINE	O	Данные файла не доступны немедленно. Этот атрибут указывает на то, что данные файла физически перемещены в автономное хранилище. Этот атрибут используется Remote Storage, которое является иерархическим программным обеспечением для управления хранилищем. Приложения не должны произвольно изменять этот атрибут.
FILE_ATTRIBUTE_READONLY	R	Файл, доступный только для чтения. Приложения могут читать

		файл, но не могут записывать в него или удалять его. Этот атрибут не учитывается в каталогах.
FILE_ATTRIBUTE_SYSTEM	S	Файл или каталог, часть которого или который операционная система использует исключительно.
FILE_ATTRIBUTE_TEMPORARY	T	Файл, который используется для временного хранения. Файловые системы избегают записи данных обратно в массовое хранилище, если доступно достаточно кэш-памяти, поскольку обычно приложение удаляет временный файл после закрытия дескриптора. В этом сценарии система может полностью избежать записи данных. В противном случае данные записываются после закрытия дескриптора.

nAttributesRemove (необязательно)

При передаче параметра «ncAttributes» он должен иметь числовое значение, и эти атрибуты добавляются к текущим атрибутам файла. Атрибуты, указанные в этом параметре, удаляются из атрибутов файла.

nMaxRecursion (необязательно)

по умолчанию = -1

-1 = без рекурсии

0 = рекурсия во все подкаталоги

>0 = рекурсия в х подкаталогов

Возвращаемое значение

Количество измененных файлов.

Смотрите также**Ссылки**

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)
[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileTimes](#)

Используемые функции WinApi

[FindFirstFile](#)
[FindNextFile](#)
[FindClose](#)
[SetFileAttributes](#)

SetFileTimes

Устанавливает дату и время создания, последнего доступа или последнего изменения указанного файла или каталога.

```
SetFileTimes( cFileName , tCreationTime [, tLastAccessTime [,  
tLastWriteTime [, bUTCTimes ]]])
```

Параметры

cFileName

Имя файла, для которого необходимо изменить время файла.

Примечание

Это может быть относительный путь или только имя файла, функция VFP [FULLPATH](#) используется для получения полного имени файла. Используется обычный путь поиска VFP ([SET PATH](#)).

tCreationTime

Значение даты и времени, которое сохраняется как новое время создания файла или каталога.

Примечание

Вы также можете передать NULL для любого параметра времени, если его не нужно устанавливать.

tLastAccessTime

Значение даты и времени, которое сохраняется как новое время последнего доступа к файлу или каталогу.

tLastWriteTime

Значение даты и времени, которое сохраняется как новое время последней записи файла или каталога.

bUTCTimes (необязательно)

по умолчанию = .F.

Если .T., функция предполагает, что переданное время указано в формате UTC, в противном случае время преобразуется в формат UTC, поскольку время файлов хранится в файловой системе в формате UTC.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ADirectoryInfo](#)
[ADirEx](#)
[ADriveInfo](#)
[AFileAttributes](#)
[AFileAttributesEx](#)
[CancelFileChange](#)
[CompareFileTimes](#)
[CopyFileEx](#)
[DeleteDirectory](#)
[DeleteFileEx](#)
[FindFileChange](#)
[GetFileAttributesEx2](#)

[GetFileOwner](#)
[GetFileSizeEx2](#)
[GetFileTimes](#)
[GetLongPathNameEx](#)
[GetShortPathNameEx](#)
[MoveFileEx](#)
[SetFileAttributesEx](#)

Используемые функции WinApi

[SetFileTime](#)
[CreateFile](#)
[CloseHandle](#)

SetSystemTimeEx

Устанавливает текущее системное время и дату.

```
SetSystemTimeEx( tTime [, bUTCTime ] )
```

Параметры

tTime

Значение даты и времени.

bUTCTime (необязательно)

по умолчанию = .F.

Если .T., то предполагается, что переданная дата и время соответствуют времени UTC.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FI](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

[UTC2DT](#)

Используемые функции WinApi

[SetLocalTime](#)

[SetSystemTime](#)

SHBrowseFolder

Отображает диалоговое окно, позволяющее пользователю выбрать папку Shell.

```
SHBrowseFolder( cTitle , nFlags , @cPath [, cRootPath [, cCallback ]])
```

Параметры

cTitle

Строка, которая отображается над элементом управления видом дерева в диалоговом окне. Эта строка может использоваться для указания инструкций пользователю.

nFlags

Флаги, которые определяют параметры диалогового окна. Этот параметр может быть 0 или комбинацией следующих значений.

Флаг	Значение
BIF_RETURNONLYFSDIRS	Возвращает только каталоги файловой системы. Если пользователь выбирает папки, которые не являются частью файловой системы, кнопка ОК становится серой. Примечание. Кнопка ОК остается включенной для элементов "\\server", а также "\\server\share" и элементов каталога. Однако, если пользователь выбирает элемент "\\server", передача PIDL, возвращаемого SHBrowseForFolder , в SHGetPathFromIDList завершается неудачей.
BIF_DONTGOBELOWDOMAIN	Не включайте сетевые папки ниже уровня домена в элемент управления древовидной структурой диалогового окна.
BIF_STATUSTEXT	Включить область статуса в диалоговое окно. Функция обратного вызова может задать текст статуса, отправляя сообщения в диалоговое окно. Этот флаг не поддерживается, если указано BIF_NEWDIALOGSTYLE.
BIF_RETURNFSANCESTORS	Возвращать только предков файловой системы. Предок — это подпапка, которая находится ниже корневой папки в иерархии пространства имен. Если пользователь выбирает предка корневой папки, который не является частью файловой системы, кнопка ОК становится серой.
BIF_EDITBOX	Включите элемент управления редактированием в диалоговое окно обзора, который позволит пользователю вводить имя элемента.
BIF_VALIDATE	BrowseCallbackProc приложения с сообщением BFFM_VALIDATEFAILED. Этот флаг игнорируется, если BIF_EDITBOX не указан.
BIF_NEWDIALOGSTYLE	Используйте новый пользовательский интерфейс. Установка этого флага предоставляет пользователю большее диалоговое окно, размер которого можно изменять. Диалоговое окно имеет несколько новых возможностей, включая: возможность перетаскивания в диалоговом окне, изменение порядка, контекстные меню, новые папки, удаление и другие команды контекстного меню.
BIF_BROWSEINCLUDEURLS	Диалоговое окно обзора может отображать URL-адреса. Также должны быть установлены флаги BIF_USENEWUI и BIF_BROWSEINCLUDEFILES. Если какой-либо из этих

	трех флагов не установлен, диалоговое окно браузера отклоняет URL-адреса. Даже если эти флаги установлены, диалоговое окно обзора отображает URL-адреса только в том случае, если папка, содержащая выбранный элемент, поддерживает URL-адреса. Когда вызывается метод папки <code>IShellFolder::GetAttributesOf</code> для запроса атрибутов выбранного элемента, папка должна установить флаг атрибута <code>SFGAO_FOLDER</code> . В противном случае диалоговое окно обзора не отобразит URL-адрес.
<code>BIF_USENEWUI</code>	Используйте новый пользовательский интерфейс, включая поле редактирования. Этот флаг эквивалентен <code>BIF_EDITBOX</code> <code>BIF_NEWDIALOGSTYLE</code> .
<code>BIF_UAHINT</code>	В сочетании с <code>BIF_NEWDIALOGSTYLE</code> добавляет подсказку по использованию в диалоговое окно вместо поля редактирования. <code>BIF_EDITBOX</code> переопределяет этот флаг.
<code>BIF_NONEWFOLDERBUTTON</code>	Не включайте кнопку «Новая папка» в диалоговое окно обзора.
<code>BIF_NOTRANSLATETARGETS</code>	Если выбранный элемент является ярлыком, возвращается PIDL самого ярлыка, а не его цели.
<code>BIF_BROWSEFORCOMPUTER</code>	Возвращает только компьютеры. Если пользователь выбирает что-то, кроме компьютера, кнопка ОК становится серой.
<code>BIF_BROWSEFORPRINTER</code>	Разрешить только выбор принтеров. Если пользователь выбирает что-либо, кроме принтера, кнопка ОК становится серой. В Windows XP и более поздних системах лучше всего использовать диалоговое окно в стиле Windows XP, установив корневой каталог диалогового окна в папку «Принтеры и факсы» (<code>CSIDL_PRINTERS</code>).
<code>BIF_BROWSEINCLUDEFILES</code>	В диалоговом окне обзора отображаются как файлы, так и папки.
<code>BIF_SHAREABLE</code>	Диалоговое окно обзора может отображать общие ресурсы на удаленных системах. Это предназначено для приложений, которые хотят предоставить удаленные общие ресурсы на локальной системе. Также должен быть установлен флаг <code>BIF_NEWDIALOGSTYLE</code> .
<code>BIF_BROWSEFILEJUNCTIONS</code>	Windows 7 и более поздние версии. Разрешить просмотр папок-соединений, таких как библиотека или сжатый файл с расширением имени файла .zip.

@cPath

Переменная по ссылке, в которой сохраняется выбранная папка.

cRootPath (необязательно)

по умолчанию = NULL

Местоположение корневой папки, с которой следует начать просмотр. В диалоговом окне отображаются только указанная папка и ее подпапки в иерархии пространства имен. Этот параметр может быть пустым или NULL, в этом случае используется корень пространства имен (папка Desktop).

cCallback (необязательно)

по умолчанию = пусто

Имя функции, определяемой приложением, которую диалоговое окно вызывает при возникновении события. Функция должна иметь следующий прототип.

```
FUNCTION BrowseCallbackProc(hwnd, uMsg, lParam)
ENDFUNC
```

Описание параметров смотрите в документации [BrowseCallbackProc](#).

Возвращаемое значение

.T. если была выбрана папка, .F. если диалог был прерван.

Смотрите также

Ссылки

[GetOpenFileNameEx](#)

[GetSaveFileNameEx](#)

[MessageBoxEx](#)

Используемые функции WinApi

[SHBrowseForFolder](#)

[SHILCreateFromPath](#)

[SHGetPathFromIDList](#)

SHCopyFiles

Копирует один или несколько файлов.

```
SHCopyFiles( cSource , cTarget , nFlags [, cTitle ,
[ cRenamedFiles ]])
```

Параметры

cSource

Одно или несколько имен файлов. Эти имена должны быть полностью определенными путями, чтобы предотвратить непредвиденные результаты. Стандартные подстановочные знаки MS-DOS, такие как "*", разрешены только в позиции имени файла. Использование подстановочного знака в другом месте строки приведет к непредсказуемым результатам. Каждое имя файла должно заканчиваться одним символом [CHR](#)(0).

Если параметр nFlags содержит FOF_SOURCE_CURSOR или FOF_SOURCE_ARRAY, строка рассматривается как имя курсора или массива.

FOF_SOURCE_CURSOR по умолчанию использует поле "filename" курсора, но также возможна передача полного имени, например "mycursor.file". FOF_SOURCE_ARRAY всегда использует первый столбец массива.

cTarget

Имя файла назначения или каталога. Подстановочные знаки не допускаются. Их использование приведет к непредсказуемым результатам.

cTarget должен соответствовать следующим спецификациям: Подстановочные знаки не поддерживаются. Он может указывать несуществующие каталоги назначения. В этих случаях система пытается их создать и обычно отображает диалоговое окно с вопросом к пользователю, хочет ли он создать новый каталог. Чтобы подавить это диалоговое окно и создать каталоги без вывода сообщений, установите флаг FOF_NOCONFIRMMKDIR в *nFlags*.

Он может содержать несколько имен файлов назначения, если параметр *nFlags* указывает FOF_MULTIDESTFILES.

Используйте полностью определенные пути. Использование относительных путей не запрещено, но может иметь непредсказуемые результаты.

Если параметр nFlags содержит FOF_SOURCE_CURSOR или FOF_SOURCE_ARRAY, строка рассматривается как имя курсора или массива.

nFlags (добавочный)

по умолчанию не выбрано ни одно из следующих значений:

Флаг	Описание
FOF_ALLOWUNDO	Сохраняйте информацию об отмене, если это возможно. Операции можно отменить только из того же процесса, который выполнил исходную операцию. Если rFrom не содержит полностью определенный путь и имена файлов, этот флаг игнорируется.
FOF_FILESONLY	Выполнять операцию над файлами только в том случае, если указано подстановочное имя файла (*.*)).
FOF_MULTIDESTFILES	Параметр <i>cTarget</i> указывает несколько файлов назначения (по одному для каждого исходного файла), а не один каталог, в который должны быть помещены все исходные файлы.

FOF_NOCONFIRMATION	В каждом отображаемом диалоговом окне ответить «Да для всех».
FOF_NOCONFIRMMKDIR	Не подтверждайте создание нового каталога, если операция требует его создания.
FOF_NO_CONNECTED_ELEMENTS	Shell Version 5.0. Не перемещать связанные файлы как группу. Перемещать только указанные файлы.
FOF_NOCOPYSECURITYATTRIBS	Версия оболочки 4.71. Не копируйте атрибуты безопасности файла.
FOF_NOERRORUI	Не отображать пользовательский интерфейс в случае возникновения ошибки.
FOF_NORECURSION	Работайте только в локальном каталоге. Не работайте рекурсивно в подкаталогах.
FOF_RENAMEONCOLLISION	Присвойте файлу, над которым выполняется операция перемещения, копирования или переименования, новое имя, если файл с целевым именем уже существует.
FOF_SILENT	Не отображать диалоговое окно хода выполнения.
FOF_SIMPLEPROGRESS	Отобразить диалоговое окно прогресса, но не показывать имена файлов. Если вы передаете параметр cTitle, OF_SIMPLEPROGRESS устанавливается автоматически.
FOF_SOURCE_CURSOR	Параметр cSource — это имя курсора.
FOF_SOURCE_ARRAY	Параметр cSource — это имя массива.
FOF_DEST_CURSOR	Параметр cTarget — это имя курсора.
FOF_DEST_ARRAY	Параметр cTarget — это имя массива.
FOF_MAPPING_ARRAY	Параметр cRenamedFiles — это имя массива, по умолчанию создается курсор.

cTitle (необязательно)

Заголовок диалога прогресса.

Если вы передаете этот параметр, флаг FOF_SIMPLEPROGRESS устанавливается автоматически, поскольку нельзя указать заголовок без установки этого флага.

cRenamedFiles (необязательно)

Применимо только при передаче FOF_RENAMEONCOLLISION в параметре nFlags.

Имя курсора или массива, в котором будут сохранены старые и новые имена файлов.

Возвращаемое значение

1, если все файлы были успешно скопированы, 0, если операция была прервана, в противном случае код ошибки из API SHFileOperation .

Смотрите также**Ссылки**

[SHDeleteFiles](#)
[SHGetShellItem](#)
[SHMoveFiles](#)
[SHRenameFiles](#)
[SHSpecialFolder](#)

Используемые функции WinApi

[SHFileOperation](#)

SHDeleteFiles

Удаляет один или несколько файлов.

```
SHDeleteFiles( cFiles , nFlags [, cTitle ])
```

Параметры

cFiles

Одно или несколько имен файлов. Эти имена должны быть полностью определенными путями, чтобы предотвратить непредвиденные результаты. Стандартные подстановочные знаки MS-DOS, такие как "*", разрешены только в позиции имени файла. Использование подстановочного знака в другом месте строки приведет к непредсказуемым результатам. Каждое имя файла должно заканчиваться одним символом [CHR](#)(0).

Если параметр nFlags содержит FOF_SOURCE_CURSOR или FOF_SOURCE_ARRAY, строка рассматривается как имя курсора или массива.

FOF_SOURCE_CURSOR по умолчанию использует поле "filename" курсора, но также возможна передача полного имени, например "mycursor.file". FOF_SOURCE_ARRAY всегда использует первый столбец массива.

nFlags (добавочный)

по умолчанию не выбрано ни одно из следующих значений:

Флаг	Описание
FOF_ALLOWUNDO	Сохраняйте информацию об отмене, если это возможно. Операции можно отменить только из того же процесса, который выполнил исходную операцию. Если rFrom не содержит полностью определенный путь и имена файлов, этот флаг игнорируется.
FOF_FILESONLY	Выполнять операцию над файлами только в том случае, если указано подстановочное имя файла (*.*)
FOF_MULTIDESTFILES	Параметр cTarget указывает несколько файлов назначения (по одному для каждого исходного файла), а не один каталог, в который должны быть помещены все исходные файлы.
FOF_NOCONFIRMATION	В каждом отображаемом диалоговом окне ответьте «Да для всех».
FOF_NOCONFIRMMKDIR	Не подтверждайте создание нового каталога, если операция требует его создания.
FOF_NO_CONNECTED_ELEMENTS	Shell Version 5.0. Не перемещать связанные файлы как группу. Перемещать только указанные файлы.
FOF_NOCOPYSECURITYATTRIBS	Версия оболочки 4.71. Не копируйте атрибуты безопасности файла.
FOF_NOERRORUI	Не отображать пользовательский интерфейс в случае возникновения ошибки.
FOF_NORECURSION	Работайте только в локальном каталоге. Не работайте рекурсивно в подкаталогах.
FOF_RENAMEONCOLLISION	Присвойте файлу, над которым выполняется операция перемещения, копирования или переименования, новое имя, если файл с целевым именем уже существует.
FOF_SILENT	Не отображать диалоговое окно хода выполнения.
FOF_SIMPLEPROGRESS	Отобразить диалоговое окно прогресса, но не

показывать имена файлов. Если вы передаете параметр `cTitle`, `FOF_SIMPLEPROGRESS` устанавливается автоматически.

cTitle (необязательно)

Заголовок диалога прогресса.

Если вы передаете этот параметр, флаг `FOF_SIMPLEPROGRESS` устанавливается автоматически, поскольку нельзя указать заголовок без установки этого флага.

Возвращаемое значение

1, если все файлы были успешно удалены, 0, если операция была прервана, в противном случае код ошибки из API [SHFileOperation](#).

Замечания

`SHDeleteFiles` навсегда удаляет файл, если вы не установите флаг `FOF_ALLOWUNDO` в параметре *nFlags*. Установка этого флага отправляет файл в корзину. Если вы хотите просто удалить файл и гарантировать, что он не будет помещен в корзину, используйте [DeleteFileEx](#).

Смотрите также**Ссылки**

[SHCopyFiles](#)
[SHGetShellItem](#)
[SHMoveFiles](#)
[SHRenameFiles](#)
[SHSpecialFolder](#)

Используемые функции WinApi

[SHFileOperation](#)

SHGetShellItem

Возвращает скриптовый прокси-объект через интерфейс IShellItem2.

```
SHGetShellItem( cFilePath )
```

Параметры

cFilePath

Путь к имени файла.

Возвращаемое значение

Объект

Замечания

Возвращаемый объект поддерживает следующие методы:

- GetProperty(cPropertyName) — извлекает указанное свойство.
- GetPropertyDescriptionList(cPropGroup (необязательно)) — возвращает объект для перечисления доступных свойств (IPropertyDescriptionList).
- GetPropertyStore() — возвращает хранилище свойств (IPropertyStore) для чтения/записи свойств.

IPropertyDescriptionList поддерживает следующие методы:

- GetAt(lnIndex) — возвращает описание свойства (IPropertyDescription) по переданному индексу (отсчет начинается с нуля).
- GetCount() — возвращает количество описаний свойств. Объект также поддерживает интерфейс перечисления — можно закодировать цикл FOR EACH loObj IN loPropertyDescriptionList для итерации по всем свойствам.

IPropertyDescription поддерживает следующие методы:

- GetCanonicalName() — возвращает полностью классифицированное имя свойства
- GetPropertyType() — возвращает базовый тип свойства
- GetDisplayName() — возвращает локализованное отображаемое имя

IPropertyStore поддерживает следующие методы:

- GetValue(lcProperty) — считывает указанное свойство
- SetValue(lcProperty, vValue) — записывает новое значение в указанное свойство
- Commit() — фиксирует изменения в файле

Если для чтения/записи нескольких свойств эффективнее использовать объект IPropertyStore, обычная функция GetProperty() в IShellItem2 будет открывать и закрывать файл для каждого вызова.

Пример

См. «shellitem.prg» в примере проекта.

Смотрите также

Ссылки

[SHCopyFiles](#)
[SHDeleteFiles](#)
[SHMoveFiles](#)
[SHRenameFiles](#)
[SHSpecialFolder](#)

Используемые функции WinApi

[SHCreateItemFromParsingName](#)

SHMoveFiles

Перемещает один или несколько файлов.

```
SHMoveFiles( cSource , cTarget , nFlags [, cTitle ,
[ cRenamedFiles ]])
```

Параметры

cSource

Одно или несколько имен файлов. Эти имена должны быть полностью определенными путями, чтобы предотвратить непредвиденные результаты. Стандартные подстановочные знаки MS-DOS, такие как "*", разрешены только в позиции имени файла. Использование подстановочного знака в другом месте строки приведет к непредсказуемым результатам. Каждое имя файла должно заканчиваться одним символом [CHR](#)(0).

Если параметр nFlags содержит FOF_SOURCE_CURSOR или FOF_SOURCE_ARRAY, строка рассматривается как имя курсора или массива.

FOF_SOURCE_CURSOR по умолчанию использует поле "filename" курсора, но также возможна передача полного имени, например "mycursor.file". FOF_SOURCE_ARRAY всегда использует первый столбец массива.

cTarget

Имя файла назначения или каталога. Подстановочные знаки не допускаются. Их использование приведет к непредсказуемым результатам.

cTarget должен соответствовать следующим спецификациям: Подстановочные знаки не поддерживаются. Он может указывать несуществующие каталоги назначения. В этих случаях система пытается их создать и обычно отображает диалоговое окно с вопросом к пользователю, хочет ли он создать новый каталог. Чтобы подавить это диалоговое окно и создать каталоги без вывода сообщений, установите флаг FOF_NOCONFIRMMKDIR в *nFlags*.

Он может содержать несколько имен файлов назначения, если параметр *nFlags* указывает FOF_MULTIDESTFILES.

Используйте полностью определенные пути. Использование относительных путей не запрещено, но может иметь непредсказуемые результаты.

Если параметр nFlags содержит FOF_SOURCE_CURSOR или FOF_SOURCE_ARRAY, строка рассматривается как имя курсора или массива.

nFlags (добавочный)

по умолчанию не выбрано ни одно из следующих значений:

Флаг	Описание
FOF_ALLOWUNDO	Сохраняйте информацию об отмене, если это возможно. Операции можно отменить только из того же процесса, который выполнил исходную операцию. Если rFrom не содержит полностью определенный путь и имена файлов, этот флаг игнорируется.
FOF_FILESONLY	Выполнять операцию над файлами только в том случае, если указано подстановочное имя файла (*.*)).
FOF_MULTIDESTFILES	Параметр <i>cTarget</i> указывает несколько файлов назначения (по одному для каждого исходного файла), а не один каталог, в который должны быть помещены все исходные файлы.

FOF_NOCONFIRMATION	В каждом отображаемом диалоговом окне ответьте «Да для всех».
FOF_NOCONFIRMMKDIR	Не подтверждайте создание нового каталога, если операция требует его создания.
FOF_NO_CONNECTED_ELEMENTS	Shell Version 5.0. Не перемещать связанные файлы как группу. Перемещать только указанные файлы.
FOF_NOCOPYSECURITYATTRIBS	Версия оболочки 4.71. Не копируйте атрибуты безопасности файла.
FOF_NOERRORUI	Не отображать пользовательский интерфейс в случае возникновения ошибки.
FOF_NORECURSION	Работайте только в локальном каталоге. Не работайте рекурсивно в подкаталогах.
FOF_RENAMEONCOLLISION	Присвойте файлу, над которым выполняется операция перемещения, копирования или переименования, новое имя, если файл с целевым именем уже существует.
FOF_SILENT	Не отображать диалоговое окно хода выполнения.
FOF_SIMPLEPROGRESS	Отобразить диалоговое окно прогресса, но не показывать имена файлов. Если вы передаете параметр cTitle, FOF_SIMPLEPROGRESS устанавливается автоматически.
FOF_SOURCE_CURSOR	Параметр cSource — это имя курсора.
FOF_SOURCE_ARRAY	Параметр cSource — это имя массива.
FOF_DEST_CURSOR	Параметр cTarget — это имя курсора.
FOF_DEST_ARRAY	Параметр cTarget — это имя массива.
FOF_MAPPING_ARRAY	Параметр cRenamedFiles — это имя массива, по умолчанию создается курсор.

cRenamedFiles (необязательно)

Применимо только при передаче FOF_RENAMEONCOLLISION в параметре nFlags. Имя курсора или массива, в котором будут сохранены старые и новые имена файлов.

Возвращаемое значение

1, если все файлы были успешно перемещены, 0, если операция была прервана, в противном случае код ошибки из API [SHFileOperation](#).

Смотрите также

Ссылки

[SHCopyFiles](#)
[SHDeleteFiles](#)
[SHGetShellItem](#)
[SHRenameFiles](#)
[SHSpecialFolder](#)

Используемые функции WinApi

[SHFileOperation](#)

Short2Str

Преобразует короткое знаковое значение (16-битное числовое значение) в двоичную строку.

```
Short2Str( nValue )
```

Параметры

nValue

Числовое значение в диапазоне от -32 768 до 32 767.

Возвращаемое значение

Строка, которая в двоичном виде равна переданному короткому значению.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

SHRenameFiles

Переименовывает один или несколько файлов.

```
SHRenameFiles( cSource , cTarget , nFlags [, cTitle ] )
```

Параметры

cSource

Имя файла. Имя должно быть полным путем, чтобы избежать непредвиденных результатов.

cTarget

Имя файла назначения.

nFlags (добавочный)

По умолчанию не выбрано ни одно из следующих значений:

Флаг	Описание
FOF_ALLOWUNDO	Сохраняйте информацию об отмене, если это возможно. Операции можно отменить только из того же процесса, который выполнил исходную операцию. Если rFrom не содержит полностью определенный путь и имена файлов, этот флаг игнорируется.
FOF_FILESONLY	Выполнять операцию над файлами только в том случае, если указано подстановочное имя файла (*.*)
FOF_MULTIDESTFILES	Параметр cTarget указывает несколько файлов назначения (по одному для каждого исходного файла), а не один каталог, в который должны быть помещены все исходные файлы.
FOF_NOCONFIRMATION	В каждом отображаемом диалоговом окне ответьте «Да для всех».
FOF_NOCONFIRMMKDIR	Не подтверждайте создание нового каталога, если операция требует его создания.
FOF_NO_CONNECTED_ELEMENTS	Shell Version 5.0. Не перемещать связанные файлы как группу. Перемещать только указанные файлы.
FOF_NOCOPYSECURITYATTRIBS	Версия оболочки 4.71. Не копируйте атрибуты безопасности файла.
FOF_NOERRORUI	Не отображать пользовательский интерфейс в случае возникновения ошибки.
FOF_NORECURSION	Работайте только в локальном каталоге. Не работайте рекурсивно в подкаталогах.
FOF_RENAMEONCOLLISION	Присвойте файлу, над которым выполняется операция перемещения, копирования или переименования, новое имя, если файл с целевым именем уже существует.
FOF_SILENT	Не отображать диалоговое окно хода выполнения.
FOF_SIMPLEPROGRESS	Отобразить диалоговое окно прогресса, но не показывать имена файлов. Если вы передаете параметр cTitle, FOF_SIMPLEPROGRESS устанавливается автоматически.

cTitle (необязательно)

Заголовок диалога прогресса.

Если вы передаете этот параметр, флаг FOF_SIMPLEPROGRESS устанавливается автоматически, поскольку нельзя указать заголовок без установки этого флага.

Возвращаемое значение

1, если файл был успешно переименован, 0, если операция была прервана, в противном случае код ошибки из API [SHFileOperation](#).

Замечания

Эту функцию нельзя использовать для переименования нескольких файлов одним вызовом функции. Вместо этого используйте [SHMoveFiles](#).

Смотрите также

Ссылки

[SHCopyFiles](#)
[SHDeleteFiles](#)
[SHGetShellItem](#)
[SHMoveFiles](#)
[SHSpecialFolder](#)

Используемые функции WinApi

[SHFileOperation](#)

SHSpecialFolder

Возвращает путь к специальной папке.

```
SHSpecialFolder( nFolderID , @cFolder [, bCreateFolder ] )
```

Параметры

nFolderId (добавочный)

CSIDL, идентифицирующий интересующую папку.

Одно из следующих значений.

Константа	Описание
CSIDL_ADMINTOOLS	Каталог файловой системы, который используется для хранения административных инструментов для отдельного пользователя. MMC сохранит настроенные консоли в этом каталоге, и он будет перемещаться вместе с пользователем.
CSIDL_ALTSTARTUP	Каталог файловой системы, который соответствует нелокализованной группе программ Startup пользователя. Это значение распознается в Windows Vista для обратной совместимости, но сама папка больше не существует.
CSIDL_APPDATA	Каталог файловой системы, который служит общим репозиторием для данных, специфичных для приложений. Типичный путь — C:\Documents and Settings\имя_пользователя\Application Data. Этот CSIDL поддерживается распространяемым Shfolder.dll для систем, в которых не установлена интегрированная оболочка Microsoft Internet Explorer 4.0.
CSIDL_BITBUCKET	Виртуальная папка, содержащая объекты в корзине пользователя.
CSIDL_CDBURN_AREA	Каталог файловой системы, который действует как промежуточная область для файлов, ожидающих записи на CD. Типичный путь — C:\Documents and Settings\имя_пользователя\Local Settings\Application Data\Microsoft\CD Burning.
CSIDL_COMMON_ADMINTOOLS	Каталог файловой системы, содержащий административные инструменты для всех пользователей компьютера.
CSIDL_COMMON_ALTSTARTUP	Каталог файловой системы, который соответствует нелокализованной группе программ Startup для всех пользователей. Это значение распознается в Windows Vista для обратной совместимости, но сама папка больше не существует.
CSIDL_COMMON_APPDATA	Каталог файловой системы, содержащий данные приложений для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Application Data. Эта папка используется для данных приложений, которые не являются специфичными для пользователя. Например, приложение может хранить словарь проверки орфографии, базу данных клипартов или файл журнала в папке CSIDL_COMMON_APPDATA. Эта информация не будет перемещаться и доступна любому пользователю компьютера.
CSIDL_COMMON_DESKTOPDIR	Каталог файловой системы, содержащий файлы и

ECTORY	папки, которые отображаются на рабочем столе для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Desktop.
CSIDL_COMMON_DOCUMENTS	Каталог файловой системы, содержащий документы, общие для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Documents.
CSIDL_COMMON_FAVORITES	Каталог файловой системы, который служит общим хранилищем избранных элементов, общих для всех пользователей.
CSIDL_COMMON_MUSIC	Каталог файловой системы, который служит хранилищем для музыкальных файлов, общих для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Documents\My Music.
CSIDL_COMMON_OEM_LINKS	Это значение распознается в Windows Vista для обратной совместимости, но сама папка больше не используется.
CSIDL_COMMON_PICTURES	Каталог файловой системы, который служит хранилищем для файлов изображений, общих для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Documents\My Pictures.
CSIDL_COMMON_PROGRAMS	Каталог файловой системы, содержащий каталоги для общих групп программ, которые отображаются в меню «Пуск» для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Start Menu\Programs.
CSIDL_COMMON_STARTMENU	Каталог файловой системы, содержащий программы и папки, которые отображаются в меню «Пуск» для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Start Menu.
CSIDL_COMMON_STARTUP	Каталог файловой системы, содержащий программы, которые отображаются в папке «Автозагрузка» для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Start Menu\Programs\Startup.
CSIDL_COMMON_TEMPLATES	Каталог файловой системы, содержащий шаблоны, доступные всем пользователям. Типичный путь — C:\Documents and Settings\All Users\Templates.
CSIDL_COMMON_VIDEO	Каталог файловой системы, который служит хранилищем для видеофайлов, общих для всех пользователей. Типичный путь — C:\Documents and Settings\All Users\Documents\My Videos.
CSIDL_COMPUTERSNEARME	Папка, представляющая другие компьютеры в вашей рабочей группе.
CSIDL_CONNECTIONS	Виртуальная папка, представляющая сетевые подключения, содержащая сетевые и коммутируемые соединения.
CSIDL_CONTROLS	Виртуальная папка, содержащая значки приложений Панели управления.
CSIDL_COOKIES	Каталог файловой системы, который служит общим репозиторием для интернет-cookies. Типичный путь — C:\Documents and Settings\имя_пользователя\Cookies.
CSIDL_DESKTOP	Виртуальная папка, представляющая рабочий стол Windows, корень пространства имен.
CSIDL_DESKTOPDIRECTORY	Каталог файловой системы, используемый для

	физического хранения файловых объектов на рабочем столе (не путать с самой папкой рабочего стола). Типичный путь — C:\Documents and Settings\имя_пользователя\Рабочий стол.
CSIDL_DRIVES	Виртуальная папка, представляющая Мой компьютер, содержащая все на локальном компьютере: устройства хранения, принтеры и Панель управления. Папка также может содержать подключенные сетевые диски.
CSIDL_FAVORITES	Каталог файловой системы, который служит общим хранилищем для избранных элементов пользователя. Типичный путь — C:\Documents and Settings\имя_пользователя\Избранное.
CSIDL_FONTS	Виртуальная папка, содержащая шрифты. Типичный путь — C:\Windows\Fonts.
CSIDL_HISTORY	Каталог файловой системы, который служит общим хранилищем элементов истории Интернета.
CSIDL_INTERNET	Виртуальная папка для Internet Explorer.
CSIDL_INTERNET_CACHE	Каталог файловой системы, который служит общим репозиторием для временных файлов Интернета. Типичный путь — C:\Documents and Settings\имя_пользователя\Local Settings\Temporary Internet Files.
CSIDL_LOCAL_APPDATA	Каталог файловой системы, который служит хранилищем данных для локальных (неперемещаемых) приложений. Типичный путь — C:\Documents and Settings\имя_пользователя\Local Settings\Application Data.
CSIDL_MYDOCUMENTS	Виртуальная папка, представляющая элемент рабочего стола «Мои документы». Это значение эквивалентно CSIDL_PERSONAL.
CSIDL_MYMUSIC	Каталог файловой системы, который служит общим репозиторием для музыкальных файлов. Типичный путь — C:\Documents and Settings\User\My Documents\My Music.
CSIDL_MYPICTURES	Каталог файловой системы, который служит общим репозиторием для файлов изображений. Типичный путь — C:\Documents and Settings\имя_пользователя\Мои документы\Мои рисунки.
CSIDL_MYVIDEO	Каталог файловой системы, который служит общим репозиторием для видеофайлов. Типичный путь — C:\Documents and Settings\имя_пользователя\Мои документы\Мои видео.
CSIDL_NETHOOD	Каталог файловой системы, содержащий объекты ссылок, которые могут существовать в виртуальной папке My Network Places. Это не то же самое, что CSIDL_NETWORK, который представляет корень сетевого пространства имен. Типичный путь — C:\Documents and Settings\username\NetHood.
CSIDL_NETWORK	Виртуальная папка, представляющая сетевое окружение, корень иерархии сетевого пространства имен.
CSIDL_PERSONAL	Виртуальная папка, представляющая элемент рабочего стола «Мои документы». Эквивалентно CSIDL_MYDOCUMENTS. До версии 6.0. Каталог файловой системы,

	используемый для физического хранения общего репозитория документов пользователя. Типичный путь — C:\Documents and Settings\имя_пользователя\Мои документы. Его следует отличать от виртуальной папки «Мои документы» в пространстве имен. Чтобы получить доступ к этой виртуальной папке, используйте SHGetFolderLocation, который возвращает ITEMIDLIST для виртуального расположения, или обратитесь к методике, описанной в разделе Управление файловой системой.
CSIDL_PRINTERS	Виртуальная папка, содержащая установленные принтеры.
CSIDL_PRINTHOOD	Каталог файловой системы, содержащий объекты ссылок, которые могут существовать в виртуальной папке Printers. Типичный путь — C:\Documents and Settings\имя_пользователя\PrintHood.
CSIDL_PROFILE	Папка профиля пользователя. Типичный путь — C:\Users\username. Приложения не должны создавать файлы или папки на этом уровне; они должны размещать свои данные в местах, на которые ссылаются CSIDL_APPDATA или CSIDL_LOCAL_APPDATA. Однако, если вы создаете новую известную папку, корень профиля, на который ссылается CSIDL_PROFILE, будет подходящим.
CSIDL_PROGRAM_FILES	Папка Program Files. Типичный путь — C:\Program Files.
CSIDL_PROGRAM_FILES_COMMON	Папка для компонентов, которые являются общими для приложений. Типичный путь — C:\Program Files\Common. Действительно только для Windows XP.
CSIDL_PROGRAMS	Каталог файловой системы, содержащий группы программ пользователя (которые сами по себе являются каталогами файловой системы). Типичный путь — C:\Documents and Settings\имя_пользователя\Главное меню\Программы.
CSIDL_RECENT	Каталог файловой системы, содержащий ярлыки для последних использованных пользователем документов. Типичный путь — C:\Documents and Settings\имя_пользователя\Мои недавние документы. Чтобы создать ярлык в этой папке, используйте SHAddToRecentDocs. Помимо создания ярлыка, эта функция обновляет список последних документов Shell и добавляет ярлык в подменю Мои недавние документы меню Пуск.
CSIDL_RESOURCES	Windows Vista. Каталог файловой системы, содержащий данные ресурсов. Типичный путь — C:\Windows\Resources.
CSIDL_SENDTO	Каталог файловой системы, содержащий пункты меню «Отправить». Типичный путь — C:\Documents and Settings\имя_пользователя\SendTo.
CSIDL_STARTMENU	Каталог файловой системы, содержащий элементы меню «Пуск». Типичный путь — C:\Documents and Settings\имя_пользователя\Меню «Пуск».
CSIDL_STARTUP	Каталог файловой системы, который соответствует группе программ автозагрузки пользователя. Система запускает эти программы при каждом входе пользователя в систему. Типичный путь — C:\Documents and Settings\имя_пользователя\Главное

	меню\Программы\Автозагрузка.
CSIDL_SYSTEM	Папка Windows System. Типичный путь — C:\Windows\System32.
CSIDL_TEMPLATES	Каталог файловой системы, который служит общим репозиторием для шаблонов документов. Типичный путь — C:\Documents and Settings\имя_пользователя\Шаблоны.
CSIDL_WINDOWS	Каталог Windows или SYSROOT. Соответствует переменным среды %windir% или %SYSTEMROOT%. Типичный путь — C:\Windows.

@cFolder

Переменная по ссылке, в которой хранится запрошенный путь к папке.

bCreateFolder (необязательно)

по умолчанию = .F.

Указывает, следует ли создавать папку, если она еще не существует. Если это значение равно .T., папка создается.

Возвращаемое значение

.T. если операция прошла успешно, .F. в противном случае.

Смотрите также**Ссылки**

[SHCopyFiles](#)
[SHDeleteFiles](#)
[SHGetShellItem](#)
[SHMoveFiles](#)
[SHRenameFiles](#)

Используемые функции WinApi

[SHGetSpecialFolderPath](#)

SizeOfMem

Возвращает размер блока памяти, выделенного [AllocMem](#).

```
SizeOfMem( nAddress )
```

Параметры

nAddress

Указатель на блок памяти, возвращаемый [AllocMem](#).

Возвращаемое значение

Если функция завершается успешно, возвращаемое значение — запрошенный размер выделенного блока памяти в байтах.

Если функция завершается неудачей, возвращаемое значение равно -1.

Если параметр *nAddress* ссылается на выделение кучи, которое не находится в библиотечной куче, поведение функции SizeOfMem не определено.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[UnlockHGlobal](#)
[ValidateMem](#)

Используемые функции WinApi

[HeapSize](#)

SQLCancelEx

Освобождает подготовленный оператор SQL.

```
SQLCancelEx( nStatement )
```

Параметры

nStatement

Подготовленный дескриптор оператора, возвращенный из [SQLPrepareEx](#).

Возвращаемое значение

1, если функции выполнены успешно.
-1, если функция не выполнена.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#) [SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLFreeHandle](#)

SQLColumnPrivilegesEx

Создает курсор столбцов и связанных с ними привилегий для указанной таблицы.

```
SQLColumnPrivilegesEx( nConnection , cCatalog , cSchema , cTable ,  
cColumn , cCursor [, nFlags ])
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cTable

Таблица, тип таблицы или имя представления в удаленной базе данных действуют как фильтр.
Либо допустимое имя схемы, либо пустая строка или .NULL. .

cCursor

Имя создаваемого курсора.

***nFlags* (необязательно)**

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

Возвращает 1 в случае успеха, -1 в случае ошибки.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#) [SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)

[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)

[SQLFetch](#)

[SQLFreeHandle](#)

SQLColumnsEx

Создает курсор имен столбцов в указанных таблицах.

```
SQLColumnsEx( nConnection , cCatalog , cSchema , cTable , cColumn ,  
cCursor [, nFlags [, nScope ]])
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cTable

Таблица, тип таблицы или имя представления в удаленной базе данных действуют как фильтр.
Либо допустимое имя схемы, либо пустая строка или .NULL. .

cCursor

Имя создаваемого курсора.

nFlags (необязательно)

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

Возвращает 1 в случае успеха, -1 в случае ошибки.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLPrimaryKeysEx](#) [SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)

[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)

[SQLFetch](#)

[SQLFreeHandle](#)

SQLCreateTableTypeCursor

Создает пустой курсор указанного зарегистрированного типа таблицы SQL Server.

Вы можете заполнить этот курсор, а затем передать ему вызов [SQLExecEx](#) для параметров TVP (параметры с табличными значениями).

```
SQLCreateTableTypeCursor( nConnection , cCatalog , cSchema ,  
cType , cCursor )
```

Параметры

nConnection

Действительный дескриптор соединения ODBC.

cCatalog

Каталожное наименование типа.

cSchema

Имя схемы типа.

cType

Название типа таблицы.

cCursor

Имя создаваемого курсора.

Возвращаемое значение

Возвращает всегда 1.

Замечания

Передайте те же параметры для параметров cCatalog, cSchema и cType, которые были переданы функции [SQLRegisterTableType](#).

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

SQLExecEx

Расширенный [SQLEXEC](#) отправляет SQL-оператор в источник данных, где он обрабатывается.

```
SQLExecEx( nConnection | nStatement , cSQL [, cCursors |
cVariables [, cArrayName [, nFlags [, cCursorSchema [,
cParamSchema [, cCallback [, nCallbackinterval [, cStatusCursor [,
nBatchSize , nRowFetchSize ]]]]]]]])
```

Параметры

nConnection | nStatement

Действительное соединение ODBC или подготовленный дескриптор оператора, возвращенный [SQLPrepareEx](#).

cSQL

Оператор SQL для выполнения.

Параметры могут быть встроены в оператор SQL с помощью следующих escape-последовательностей:

Тип параметра	Escape-последовательность
Нормальный входной параметр	?{m.lcVariableName}
Входной/выходной параметр	?@{m.lcVariableName}
Только выходной параметр	?&{m.lcVariableName}
Входной параметр потока	?\${m.lcVariableName}
Параметры потока ввода/вывода	?@\${m.lcVariableName}

Любые строки, заключенные в одинарные или двойные кавычки, будут переданы как есть.

cCursors | cVariables (необязательно)

Указывает одно или несколько имен курсоров, разделенных запятой, например, "yourCursor1, yourCursor2"

Если вы опустите этот параметр или передадите пустую строку, имена будут созданы автоматически, как в [SQLEXEC](#), первый курсор будет назван "sqlresult", а для каждого дополнительного курсора будет добавлен номер набора результатов (sqlresult2, sqlresult3).

Если вы передадите SQLEXEC_DEST_VARIABLE в параметре *nFlags*, значение будет интерпретировано как список имен переменных или полей, разделенных запятыми, в котором данные первой строки набора результатов сохраняются в последовательном порядке, то есть столбец № 1 сохраняется в переменной/поле № 1, 2 in2 и т. д.

cArray (необязательно)

Имя массива, в котором хранится количество возвращенных/ удаленных/ обновленных/ вставленных строк для каждого оператора.

Столбец	Значение
1	Имя курсора или пустая строка, если набор результатов не сгенерирован.
2	Количество возвращенных, удаленных, обновленных или вставленных строк.

Это точно такое же поведение, которое было введено в VFP9. Если указать SQLEXEC_STORE_INFO в параметре *nFlags*, дополнительная информация будет сохранена в первом столбце массива.

nFlags (необязательный, дополнительный)

по умолчанию = SQLECEX_DEST_CURSOR | SQLECEX_CALLBACK_PROGRESS | SQLECEX_CALLBACK_INFO

Если параметр *nFlags* опущен или передано значение 0, будет использоваться значение по умолчанию.

Флаг	Описание
SQLECEX_DEST_CURSOR	Сохраните набор результатов в одном или нескольких курсорах VFP.
SQLECEX_DEST_VARIABLE	Сохраните набор результатов в переменных/полях VFP. Если вы передадите этот флаг, то в переменных/полях будут сохранены только значения первой строки набора результатов.
SQLECEX_REUSE_CURSOR	Повторно используйте курсор, если он уже существует, если курсор существует, он ЗАП'ится перед сохранением в нем новых записей. Передача SQLECEX_REUSE_CURSOR включает SQLECEX_DEST_CURSOR, вам не нужно передавать оба.
SQLECEX_NATIVE_SQL	Не пытайтесь разобрать переданный оператор SQL на параметры, просто передайте его в источник данных как есть. Конечно, вы не сможете встроить какие-либо параметры, если установите этот флаг.
SQLECEX_CALLBACK_PROGRESS	Обратный вызов функции, переданной в параметре <i>cCallback</i> при извлечении строк. Если параметр <i>cCallback</i> пропущен, этот флаг игнорируется.
SQLECEX_CALLBACK_INFO	Обратный вызов функции, переданной в параметре <i>cCallback</i> , если получена дополнительная информация от бэкэнда (например, из операторов PRINT или RAISERROR в хранимой процедуре).
SQLECEX_STORE_INFO	Сохраните дополнительную информацию из бэкэнда в массиве, переданном в параметре <i>cArray</i> .
SQLECEX_APPEND_CURSOR	Повторно используйте курсор и не удаляйте его, новые записи просто добавляются. Передача SQLECEX_APPEND_CURSOR включает SQLECEX_REUSE_CURSOR и SQLECEX_DEST_CURSOR, вам не нужно передавать все.
SQLECEX_PRESERVE_RECNO	Действительно только в сочетании с SQLECEX_APPEND_CURSOR, восстанавливает указатель записи после извлечения новых строк.
SQLECEX_PARAMETER_ARRAY	Отправляет SQL-оператор для массива параметров, а не только один раз; все параметры должны быть именами полей курсора.

cCursorSchema

Синтаксис схемы такой же, как в CREATE TABLE/CURSOR, список имен столбцов и описаний типов, разделенных запятыми, например:
 "myIntCol I, myCharCol C(254) NULL, myBinaryCol C(254) NULL NOCPTRANS, myText M, myBlob W"

Примечание

Длинные имена типов, такие как «Numeric» или «Character», **НЕ** поддерживаются! Если вы запрашиваете у базы данных данные символов Unicode, по умолчанию выполняется преобразование в Ansi (вы можете управлять преобразованием типов символов глобально с помощью [VFP2CSys\(4\)](#)). Если вы хотите получить данные Unicode без изменений, вы можете указать NOCPTRANS для столбца в схеме, например, «yourUnicodeSmallText C(254) NOCPTRANS, yourUnicodeLongtext M NOCPTRANS»

Если вы преобразуете столбец Unicode в типы «W» или «Q», данные также будут в формате Unicode.

Примечание

Если схема не указана для поля или вообще не указана, применяются те же правила сопоставления типов данных, что и в SQLExec.

Кроме того, имейте в виду, что следующие настройки оцениваются при каждом вызове SQLExecEx и повлияют на сопоставление:

[CURSORGETPROP](#)('UseMemoSize',0) - изменяет ограничение размера

символьных/двоичных типов - все, что больше, будет сопоставлено полям memo/blob

[CURSORGETPROP](#)('MapVarchar',0) - сопоставление varchar с C или V

[CURSORGETPROP](#)('MapBinary',0) - сопоставление varbinary с C NOCPTRANS или Q

cParameterSchema (необязательно)

Список номеров параметров, имен и типов SQL, разделенных запятыми, в которых должны быть отправлены данные, например

"1 '@parnone' SQL_WCHAR, 2 '@parmtwo' SQL_CHAR"

&& Имя параметра должно быть заключено либо в " (двойные кавычки), либо в ' (одинарные кавычки)

```
lcDocument = STRTOFILE('someFileWithUnicodeContent.txt')
?SQLEXECEx(yourConnection,'UPDATE yourTable SET someField
= ?{someVar}, someUnicodeTextField = ?{lcDocument} WHERE
yourKey = ?{lnID}',' ',' ',0,' ','2 SQL_WCHAR')
```

&& Вы также можете передать определенный тип данных бэкэнда, указав тип SQL в числовой форме

```
?SQLEXECEx(yourConnection,'UPDATE yourTable SET someField
= ?{someVar} WHERE yourKey = ?{lnID}',' ','laResult',0,' ','1
43')
```

Примечание

Именованные параметры можно использовать только если вы вызываете хранимые процедуры и бэкэнд и драйвер ODBC поддерживают это - например, Microsoft SQL Server.

Если вы используете именованные параметры, все параметры должны быть именованы.

cCallback (необязательно)

Функция для обратного вызова при извлечении набора результатов или при извлечении оператора PRINT или RAISERROR или другой дополнительной информации бэкэнда.

Прототип функции:

```
FUNCTION SQLCallback(lnSet, lnRow, lnRowCount)
```

&& lnSet: номер набора результатов, который в данный момент
&& извлекается, или -1, если перехвачен оператор PRINT или
&& RAISERROR с низкой степенью серьезности

&& lnRow: текущая извлекаемая строка, или если lnSet равен -1,
&& то lnRow содержит сообщение/предупреждение оператора PRINT
&& или RAISERROR.

```

&& lnRowCount: либо общее количество строк для выборки
&& или 0/-1, если используемый драйвер ODBC не поддерживает
&& возврат соответствующих строк запроса.
&& Если lnSet равен -1, этот параметр не имеет значения и
&& всегда равен .F.
ENDFUNC

```

nCallbackinterval

по умолчанию = 100

Интервал, используемый для обратного вызова процедуры обратного вызова при извлечении строк. Ваша процедура обратного вызова будет вызвана один раз перед извлечением первой строки, затем для каждой n-й записи и один раз после того, как будет извлечена последняя строка.

cStatusCursor (необязательно)

Действительно только когда параметр nFlags содержит SQLECEX_PARAMETER_ARRAY. Созданный курсор будет содержать информацию об ошибке ODBC для каждого набора параметров, которые не удалось выполнить.

nBatchSize (необязательно)

по умолчанию = 16

Действительно только когда параметр nFlags содержит SQLECEX_PARAMETER_ARRAY.

Код выполнит оператор SQL несколько раз в пакетах nBatchSize.

nRowFetchSize (необязательно)

по умолчанию = 1

Установка этого параметра на значение больше 1 будет извлекать данные пакетами nRowFetchSize, это может потенциально ускорить передачу данных по сети, поскольку базовый драйвер ODBC может более эффективно управлять сетевым трафиком.

Вероятно, стоит устанавливать только при извлечении больших наборов результатов.

Вы можете поэкспериментировать с размером выборки, чтобы найти подходящее значение (например, test, 32, 64, 128, 256, 512), после определенного значения скорость извлечения не увеличится, а будет только потреблять больше локальной памяти.

Имейте в виду, что набор результатов может не содержать столбцов с длинными типами данных, например, TEXT, VARCHAR(MAX), BLOB и т. д., чтобы эта функциональность была доступна.

Возвращаемое значение

- 1: операторы SQL были выполнены успешно.
- 2: операторы SQL были выполнены, но были выданы предупреждения, которые можно получить, вызвав AERROREX.
- 1: функция не выполнена.
- 2: операция была прервана функцией обратного вызова.

Пример

Внедрение параметров.

```
LOCAL laInfo[1], laError[1]
```

```
lcSQL = "SELECT * FROM someTable WHERE someCol = ?{someVar}"
```

```

IF SQLEXECEX(yourConnection, lcSQL, 'yourResult', 'laInfo')
    ? "Курсор " + laInfo[1,1] + " содержит " + ;
      ALLTRIM( STR(laInfo[1,2]) ) + " строк"
ELSE
    AERROREX('laError')
    DISPLAY MEMORY LIKE laError
ENDIF

lcSQL = "UPDATE someTable SET someCol = ?{myCursor.someField}
WHERE pk = ?{myCursor.pk}"
IF SQLEXECEX(yourConnection, lcSQL, '', 'laInfo') > 0
    ? laInfo[1,2], " строки обновлены"
ЕЩЕ
    AERROREX('laError')
    DISPLAY MEMORY LIKE laError
ENDIF

```

Сохранение результата в переменных вместо курсора.

```

LOCAL lnSum, lnAvg
?SQLEXECEX(yourConnection, 'SELECT SUM(someCol),
AVG(someCol2) FROM yourTable', 'lnSum, lnAvg', '',
SQLEXECEX_DEST_VARIABLE)

```

Использование именованных параметров.

```

MSSQL T-SQL:
-----
CREATE PROCEDURE teststoredproc (@lnPK int = 1,
    @description varchar(8000) = 'Hello' OUTPUT,
    @thirdParam int = 2, @fourthParam varchar(4000))
)
AS
BEGIN
    PRINT 'Procedure START'
    SET @description = REPLICATE('HelloWorld',50)
    PRINT 'Procedure finished!'
END
-----

LOCAL lcMessage, lcMessage2, lcSQL
lcMessage = ''
lcMessage2 = 'Привет, SQL-сервер'
lcSQL = "{ CALL
teststoredproc(?@{lcMessage},?{lcMessage2}) }"

?SQLEXECEX(yourConnection,lcSQL,','', 'aResult',0,'','',1
"@description", 2 "@fourthParam" ')

```

Смотрите также

Ссылки

[ASQLDataSources](#)

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#) [SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)
[SQLBindParameter](#)
[SQLGetStmtAttr](#)
[SQLSetDescRec](#)
[SQLSetDescField](#)
[SQLExecDirect](#)
[SQLParamData](#)
[SQLPutData](#)
[SQLGetDiagRec](#)
[SQLNumResultCols](#)
[SQLRowCount](#)
[SQLDescribeCol](#)
[SQLColAttribute](#)
[SQLGetInfo](#)
[SQLBindCol](#)
[SQLFetch](#)
[SQLFreeStmt](#)
[SQLMoreResults](#)
[SQLFreeHandle](#)

SQLForeignKeysEx

Создает курсор внешних ключей в указанной таблице (столбцы в указанной таблице, которые ссылаются на первичные ключи в других таблицах) или курсор внешних ключей в других таблицах, которые ссылаются на первичный ключ в указанной таблице.

```
SQLForeignKeysEx( nConnection , cCatalog , cSchema , cTable ,  
cCatalog2 , cSchema2 , cTable2 , cCusor [, nFlags ] )
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cTable

Таблица, тип таблицы или имя представления в удаленной базе данных действуют как фильтр.
Либо допустимое имя схемы, либо пустая строка или .NULL. .

cCusor

Имя создаваемого курсора.

nFlags (необязательно)

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

Возвращает 1 в случае успеха, -1 в случае ошибки.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#) [SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)

[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)
[SQLFetch](#)
[SQLFreeHandle](#)

SQLGetPropEx

Расширенный [SQLGETPROP](#), извлекает атрибуты соединения ODBC.

```
SQLGetPropEx( nConnection | cCursorName , cAttribute ,
@vAttributeValue )
```

Параметры

nConnectionHandle | cCursorName

Допустимое соединение, возвращаемое [SQLCONNECT](#) / [SQLSTRINGCONNECT](#), или имя курсора, созданного удаленным соединением - [CURSORGETPROP](#) ("ConnectHandle", cCursorName), должны возвращать допустимый дескриптор соединения.

cAttribute

Одно из следующих значений.

Значение	Описание
Trace	Запросить, включена/отключена ли трассировка отладки ODBC.
TraceFile	Запросить текущий файл трассировки отладки ODBC.
Connected	Запросить статус базового соединения.
IsolationLevel	Уровень изоляции запроса.
Perfdata	Получите указатель на структуру данных производительности MS SQL Server.

@vAttributeValue

Переменная по ссылке, в которой сохраняется запрошенный атрибут соединения.

Возвращаемое значение

[AErrorEx](#) доступна дополнительная информация.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#) [SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi[SQLGetConnectAttr](#)

SQLGetTypeInfoEx

```
SQLGetTypeInfoEx( nConnection , nType , cCursor [, nFlags ])
```

Параметры

Возвращаемое значение

SQLPrepareEx

Расширенный SQLPREPARE.

Подготавливает SQL-оператор для удаленного выполнения [SQLExecEx](#)().

```
SQLPrepareEx( nConnection , cSQL [, cCursors | cVariables [,
cArrayName [, nFlags [, cCursorSchema [, cParamSchema [, cCallback
[, nCallbackinterval ]]]]]]]))
```

Параметры

nConnection

Допустимое соединение ODBC.

cSQL

Оператор SQL для выполнения.

Параметры можно встроить в оператор SQL, заключив их в фигурные скобки с префиксом "?", все, что в операторе заключено в " или ', передается как есть.

cCursors | cVariables (необязательно)

Указывает одно или несколько имен курсоров, разделенных запятой, например, "yourCursor1, yourCursor2"

Если вы опустите этот параметр или передадите пустую строку, имена будут созданы автоматически, как в [SQLEXEC](#), первый курсор будет назван "sqlresult", а для каждого дополнительного курсора будет добавлен номер набора результатов (sqlresult2, sqlresult3).

Если вы передадите SQLECEX_DEST_VARIABLE в параметре *nFlags*, значение будет интерпретировано как список имен переменных или полей, разделенных запятыми, в котором данные первой строки набора результатов сохраняются в последовательном порядке, то есть столбец № 1 сохраняется в переменной/поле № 1, 2 in2 и т. д.

cArray (необязательно)

Имя массива, в котором хранится количество возвращенных/ удаленных/ обновленных/ вставленных строк для каждого оператора.

Столбец	Значение
1	Имя курсора или пустая строка, если набор результатов не сгенерирован.
2	Количество возвращенных, удаленных, обновленных или вставленных строк.

Это точно такое же поведение, которое было введено в VFP9. Если указать SQLECEX_STORE_INFO в параметре *nFlags*, дополнительная информация будет сохранена в первом столбце массива.

nFlags (необязательный, дополнительный)

по умолчанию = SQLECEX_DEST_CURSOR | SQLECEX_CALLBACK_PROGRESS | SQLECEX_CALLBACK_INFO Если параметр

nFlags опущен или передано значение 0, будет использоваться значение по умолчанию.

Флаг	Описание
SQLECEX_DEST_CURSOR	Сохраните набор результатов в одном или нескольких курсорах VFP.
SQLECEX_DEST_VARIABLE	Сохраните набор результатов в переменных/полях VFP. Если вы передадите этот флаг, то в переменных/полях будут сохранены только значения первой строки набора

	результатов.
SQLXECEX_REUSE_CURSOR	Повторно используйте курсор, если он уже существует, если курсор существует, он ЗАП'ится перед сохранением в нем новых записей. Передача SQLXECEX_REUSE_CURSOR включает SQLXECEX_DEST_CURSOR, вам не нужно передавать оба.
SQLXECEX_NATIVE_SQL	Не пытайтесь разобрать переданный оператор SQL на параметры, просто передайте его в источник данных как есть. Конечно, вы не сможете встроить какие-либо параметры, если установите этот флаг.
SQLXECEX_CALLBACK_PROGR ESS	Обратный вызов функции, переданной в параметре <i>cCallback</i> при извлечении строк. Если параметр <i>cCallback</i> пропущен, этот флаг игнорируется.
SQLXECEX_CALLBACK_INFO	Обратный вызов функции, переданной в параметре <i>cCallback</i> , если получена дополнительная информация от бэкэнда (например, из операторов PRINT или RAISERROR в хранимой процедуре).
SQLXECEX_STORE_INFO	Сохраните дополнительную информацию из бэкэнда в массиве, переданном в параметре <i>cArray</i> .
SQLXECEX_APPEND_CURSOR	Повторно используйте курсор и не удаляйте его, новые записи просто добавляются. Передача SQLXECEX_APPEND_CURSOR включает SQLXECEX_REUSE_CURSOR и SQLXECEX_DEST_CURSOR, вам не нужно передавать все.
SQLXECEX_PRESERVE_RECNO	Действительно только в сочетании с SQLXECEX_APPEND_CURSOR, восстанавливает указатель записи после извлечения новых строк.
SQLXECEX_PARAMETER_ARR AY	Отправляет SQL-оператор для массива параметров, а не только один раз; все параметры должны быть именами полей курсора.

cCursorSchema:

Синтаксис схемы такой же, как в CREATE TABLE/CURSOR, список имен столбцов и описаний типов, разделенных запятыми, например:
 "myIntCol I, myCharCol C(254) NULL, myBinaryCol C(254) NULL NOCPTRANS, myText M, myBlob W"

Примечание

Длинные имена типов, такие как «Numeric» или «Character», **НЕ** поддерживаются! Если вы запрашиваете у базы данных данные символов Unicode, по умолчанию выполняется преобразование в Ansi. Если вы хотите получить данные Unicode без изменений, вы можете указать NOCPTRANS для столбца в схеме. например, «yourUnicodeSmallText C(254) NOCPTRANS, yourUnicodeLongtext M NOCPTRANS».

Если вы преобразуете столбец Unicode в типы «W» или «Q», данные также будут в формате Unicode.

cParameterSchema (необязательно)

Список номеров параметров, имен и типов SQL, разделенных запятыми, в которых должны быть отправлены данные, например
 "1 '@parmone' SQL_WCHAR, 2 '@parmtwo' SQL_CHAR"

&& имя параметра должно быть заключено либо в " (двойные кавычки), либо в ' (одинарные кавычки)

```
lcDocument = STRTOFILE('someFileWithUnicodeContent.txt')
?SQLEXEC( yourConnection, 'UPDATE yourTable SET someField
= ?{someVar}, someUnicodeTextField = ?{lcDocument} WHERE
yourKey = ?{lnID}', '', '', 0, '', '2 SQL_WCHAR')
```

&& Вы также можете передать определенный тип данных бэкэнда, указав тип SQL в числовой форме

```
?SQLEXEC( yourConnection, 'UPDATE yourTable SET someField
= ?{someVar} WHERE yourKey = ?{lnID}', '', 'laResult', 0, '', '1
43')
```

Примечание

Именованные параметры можно использовать только если вы вызываете хранимые процедуры и бэкэнд и драйвер ODBC поддерживают это - например, Microsoft SQL Server.

Если вы используете именованные параметры, все параметры должны быть именованы.

cCallback (необязательно)

Функция для обратного вызова при извлечении набора результатов или при извлечении оператора PRINT или RAISERROR или другой дополнительной информации бэкэнда.

Прототип функции:

```
FUNCTION SQLCallback(lnSet, lnRow, lnRowCount)
&& lnSet: номер набора результатов, который в данный момент
&& извлекается, или -1, если обнаружен оператор PRINT или
&& RAISERROR с низкой степенью серьезности

&& lnRow: текущая извлекаемая строка, или если lnSet равен -1
&& lnRow содержит сообщение/предупреждение PRINT или
&& RAISERROR оператор

&& lnRowCount: либо общее количество строк для выборки или
&& 0/-1, если используемый драйвер ODBC не поддерживает
&& возврат совпадающих строк запроса
&& если lnSet равен -1, этот параметр не имеет значения и
&& всегда равен .F.
ENDFUNC
```

nCallbackinterval

по умолчанию = 100

Интервал, используемый для обратного вызова процедуры обратного вызова при извлечении строк. Ваша процедура обратного вызова будет вызвана один раз перед извлечением первой строки, затем для каждой n-й записи и один раз после того, как будет извлечена последняя строка.

Возвращаемое значение

Подготовленный дескриптор оператора, -1, если функция не выполнена.

Смотрите также

Ссылки

[ASQLDataSources](#)

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)

[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#)
[SQLProceduresEx SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)
[SQLPrepare](#)

SQLPrimaryKeysEx

Создает курсор с именами столбцов, составляющих первичный ключ таблицы.

```
SQLPrimaryKeysEx( nConnection , cCatalog , cSchema , cTable ,  
cCursor [, nFlags ] )
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cTable

Таблица, тип таблицы или имя представления в удаленной базе данных действуют как фильтр.
Либо допустимое имя схемы, либо пустая строка или .NULL. .

cCursor

Имя создаваемого курсора.

***nFlags* (необязательно)**

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

Возвращает 1 в случае успеха, -1 в случае ошибки.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLForeignKeysEx](#) [SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)

[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)

[SQLFetch](#)

[SQLFreeHandle](#)

SQLProcedureColumnsEx

Создает курсор входных и выходных параметров, а также столбцов, составляющих результирующий набор для указанных процедур.

```
SQLProcedureColumnsEx( nConnection , cCatalog , cSchema ,  
cProcedure , cColumn , cCursor [, nFlags ])
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cCursor

Имя создаваемого курсора.

***nFlags* (необязательно)**

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

Возвращает 1 в случае успеха, -1 в случае ошибки.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#) [SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)[SQLFetch](#)[SQLFreeHandle](#)

SQLProceduresEx

Создает курсор имен процедур, хранящихся в определенном источнике данных.

```
SQLProceduresEx( nConnection , cCatalog , cSchema , cProcedure ,  
cCursor [, nFlags ] )
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cCursor

Имя создаваемого курсора.

***nFlags* (необязательно)**

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

Возвращает 1 в случае успеха, -1 в случае ошибки.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#) [SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)[SQLFetch](#)[SQLFreeHandle](#)

SQLRegisterTableType

Регистрирует тип таблицы SQL Server в библиотеке для использования в [SQLExecEx](#).

```
SQLRegisterTableType( nConnection , cCatalog , cSchema , cType [,  
nFlags ] )
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Передайте имя каталога или пустую строку для текущей базы данных типа, который необходимо зарегистрировать.

cSchema

Передайте имя схемы типа, который необходимо зарегистрировать.

cType

Имя типа таблицы, подлежащего регистрации.

***nFlags* (необязательно)**

по умолчанию: SQLREGISTERTABLETYPE_CATALOG | SQLREGISTERTABLETYPE_SCHEMA

SQLREGISTERTABLETYPE_CATALOG

SQLREGISTERTABLETYPE_SCHEMA

Управляет именем, под которым тип должен быть зарегистрирован.

Например,

SQLREGISTERTABLETYPE_CATALOG | SQLREGISTERTABLETYPE_SCHEMA результаты в
catalogname.schemaname.typename

SQLREGISTERTABLETYPE_CATALOG результаты в
catalogname..typename

SQLREGISTERTABLETYPE_SCHEMA результаты в
.schemaname.typename

Возвращаемое значение

1, если функция выполнена успешно, или -1, если функция не выполнена.

Смотрите также

Ссылки

[ASQLDataSources](#)

[ASQLDrivers](#)

[ChangeSQLDataSource](#)

[CreateSQLDataSource](#)

[DeleteSQLDataSource](#)

[SQLCancelEx](#)

[SQLExecEx](#)

[SQLGetPropEx](#)

[SQLPrepareEx](#)

[SQLSetPropEx](#)

[SQLCreateTableTypeCursor](#)

[SQLTablesEx](#)

[SQLColumnsEx](#)

[SQLPrimaryKeysEx](#) [SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)
[SQLBindCol](#)
[SQLFetch](#)
[SQLFreeHandle](#)

SQLSetPropEx

Расширенный [SQLSETPROP](#), устанавливает атрибуты соединения ODBC.

```
SQLSetPropEx( nConnection | cCursorName , cAttribute , vValue )
```

Параметры

nConnection | *cCursorName*

Допустимое соединение, возвращаемое [SQLCONNECT](#) / [SQLSTRINGCONNECT](#), или имя курсора, созданного удаленным соединением - [CURSORGETPROP](#) ("ConnectHandle", *cCursorName*), должны возвращать допустимый дескриптор соединения.

cAttribute

Одно из следующих значений.

Значение	Описание
Trace	Запуск/остановка отладочной трассировки ODBC.
TraceFile	Установите файл трассировки отладки ODCB.
Perfdata	Запуск/остановка регистрации данных производительности MS SQL Server.
PerfdataFile	Настройте файл журнала данных производительности MS SQL Server.
PerfdataLog	Регистрируйте данные о производительности MS SQL Server.

vValue

Допустимые значения для различных атрибутов:

Значение	Допустимые значения
Trace	.F. или .T. для остановки - начала отслеживания
TraceFile	Допустимое имя файла.
Perfdata	.F. или .T. для запуска/остановки регистрации данных производительности MS SQL Server.
PerfdataFile	Допустимое имя файла.
PerfdataLog	Значение не требуется, просто передайте этот атрибут, и журнал обновится.

Возвращаемое значение

[AErrorEx](#) доступна дополнительная информация.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)

[SQLPrimaryKeysEx](#) [SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLSetConnectAttr](#)

SQLSpecialColumnsEx

Создает курсор со следующей информацией о столбцах в таблице:

- оптимальный набор столбцов, который уникально идентифицирует строку в таблице.
- столбцы, которые автоматически обновляются при обновлении любого значения в строке транзакцией.

```
SQLSpecialColumnsEx( nConnection , cCatalog , cSchema , cTable ,  
nIdentifier , nScope , bNullable , cCursor [, nFlags ] )
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cTable

Таблица, тип таблицы или имя представления в удаленной базе данных действуют как фильтр.
Либо допустимое имя схемы, либо пустая строка или .NULL. .

cCursor

Имя создаваемого курсора.

nFlags (необязательно)

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

Возвращает 1 в случае успеха, -1 в случае ошибки.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#)
[SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)

[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)
[SQLFetch](#)
[SQLFreeHandle](#)

SQLStatisticsEx

```
SQLStatisticsEx( nConnection , cCatalog , cSchema , cTable ,  
cColumn , nUnique , bReserved , cCursor [, nFlags ] )
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cTable

Таблица, тип таблицы или имя представления в удаленной базе данных действуют как фильтр.
Либо допустимое имя схемы, либо пустая строка или .NULL. .

cCursor

Имя создаваемого курсора.

***nFlags* (необязательно)**

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

SQLTablePrivilegesEx

Создает курсор таблиц и привилегий, связанных с каждой таблицей.

```
SQLTablePrivilegesEx( nConnection , cCatalog , cSchema , cTable ,  
cCursor [, nFlags ] )
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cTable

Таблица, тип таблицы или имя представления в удаленной базе данных действуют как фильтр.
Либо допустимое имя схемы, либо пустая строка или .NULL. .

cCursor

Имя создаваемого курсора.

nFlags (необязательно)

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

Возвращаемое значение

Возвращает 1 в случае успеха, -1 в случае ошибки.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)
[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLTablesEx](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#) [SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi[SQLAllocHandle](#)[SQLFetch](#)[SQLFreeHandle](#)

SQLTablesEx

Создает курсор с именами таблиц, каталогов или схем, а также типами таблиц, хранящимися в определенном источнике данных.

```
SQLTablesEx( nConnection , cCatalog , cSchema , cTable ,  
cTableType , cCursor [, nFlags [, nScope ]])
```

Параметры

nConnection

Допустимое соединение ODBC.

cCatalog

Каталог удаленной базы данных, действует как фильтр.
Либо допустимое имя каталога, пустая строка или .NULL. .

cSchema

Схема в удаленной базе данных действует как фильтр.
Либо допустимое имя схемы, либо пустая строка, либо .NULL. .

cTable

Таблица, тип таблицы или имя представления в удаленной базе данных действуют как фильтр.
Либо допустимое имя схемы, либо пустая строка или .NULL. .

cTableType

Тип таблицы действует как фильтр.
TABLE, TABLE TYPE или VIEW.

cCursor

Имя создаваемого курсора.

nFlags (необязательно)

Установит атрибуты оператора перед выполнением функции.
При передаче SQL_ATTR_METADATA_ID это повлияет на обработку параметров cCatalog, cSchema и cTable.

nScope (необязательно)

При передаче устанавливает атрибут оператора SQL_SOPT_SS_NAME_SCOPE (только Microsoft SQL Server).
Когда вы хотите перечислить типы таблиц в SQL Server, передайте SQL_SS_NAME_SCOPE_TABLE_TYPE.

Возвращаемое значение

1, если функция выполнена успешно, или -1, если она не выполнена.

Смотрите также

Ссылки

[ASQLDataSources](#)
[ASQLDrivers](#)
[ChangeSQLDataSource](#)
[CreateSQLDataSource](#)
[DeleteSQLDataSource](#)
[SQLCancelEx](#)
[SQLExecEx](#)
[SQLGetPropEx](#)
[SQLPrepareEx](#)

[SQLSetPropEx](#)
[SQLRegisterTableType](#)
[SQLCreateTableTypeCursor](#)
[SQLColumnsEx](#)
[SQLPrimaryKeysEx](#)
[SQLForeignKeysEx](#) [SQLProceduresEx](#)
[SQLProcedureColumnsEx](#)
[SQLTablePrivilegesEx](#)
[SQLColumnPrivilegesEx](#)
[SQLSpecialColumnsEx](#)
[SQLVariantToValue](#)
[ValueToSQLVariant](#)

Используемые функции WinApi

[SQLAllocHandle](#)
[SQLFetch](#)
[SQLFreeHandle](#)

SQLVariantToValue

Преобразует поле типа SQL_SS_VARIANT, полученное из SQL Server через [SQLExecEx](#), в эквивалентный ему тип FoxPro.

```
SQLVariantToValue( cValue [, @nType [, @nSize [, @nScale ]]])
```

Параметры

cValue

Значение, полученное из SQL Server.

Примечание

Поля BLOB-объектов курсора должны передаваться по ссылке.

@nType (необязательно)

хранит числовые данные ODBC.

@nSize (необязательно)

сохраняет размер в байтах.

@nScale (необязательно)

хранит дробные цифры.

Возвращаемое значение

Верните преобразованное значение.

Смотрите также

Ссылки

- [ASQLDataSources](#)
- [ASQLDrivers](#)
- [ChangeSQLDataSource](#)
- [CreateSQLDataSource](#)
- [DeleteSQLDataSource](#)
- [SQLCancelEx](#)
- [SQLExecEx](#)
- [SQLGetPropEx](#)
- [SQLPrepareEx](#)
- [SQLSetPropEx](#)
- [SQLRegisterTableType](#)
- [SQLCreateTableTypeCursor](#)
- [SQLTablesEx](#)
- [SQLColumnsEx](#)
- [SQLPrimaryKeysEx](#)
- [SQLForeignKeysEx](#)
- [SQLProceduresEx](#)
- [SQLProcedureColumnsEx](#)
- [SQLTablePrivilegesEx](#)
- [SQLColumnPrivilegesEx](#)
- [SQLSpecialColumnsEx](#)
- [ValueToSQLVariant](#)

ST2DT

Преобразует структуру [SYSTEMTIME](#) в значение datetime.

```
ST2DT( nSystemTimePointer [, bToLocalTime ])
```

Параметры

nSystemTimePointer

Указатель на структуру [SYSTEMTIME](#).

bToLocalTime (необязательно)

по умолчанию = .F.

Если .T, то время файла преобразуется в местный часовой пояс.

Возвращаемое значение

Дата и время.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FI](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[Timet2DT](#)

[UTC2DT](#)

StartServiceEx

Запускает службу Windows.

```
StartServiceEx( cServiceName | nServiceHandle [, @aArguments [,  
nTimeout [, cServer [, cDatabase ]]])
```

Параметры

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

@aArguments (необязательно)

Массив строк, которые передаются в качестве аргументов сервису.

Если вы хотите опустить этот параметр, но хотите передать параметры, которые идут после него, вы можете передать NULL вместо допустимого массива.

nTimeout (необязательно)

Максимальное время ожидания в секундах, пока служба находится в состоянии SERVICE_START_PENDING.

Если вы передадите 0, функция не будет ждать, пока служба будет инициализирована, вместо этого она вернется сразу после отправки запроса на запуск.

Если вы опустите этот параметр или передадите NULL, тайм-аут будет установлен на тайм-аут по умолчанию, сообщенный службой. См. справку MSDN для члена "dwWaitHint" структуры [SERVICE_STATUS_PROCESS](#).

cServer (необязательно)

Имя сервера, на котором запущена служба.

См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.

См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

1, если служба была успешно запущена или уже работала, 0, если интервал ожидания истек до того, как служба перешла в состояние работы.

Смотрите также

Ссылки

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StopServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[StartService](#)
[QueryServiceStatus](#)

[OpenSCManager](#)[OpenService](#)

StopServiceEx

Отправляет запрос на остановку указанной службе.

```
StopServiceEx( cServiceName | nServiceHandle [, nTimeout [,  
bStopDependantServices [, cServer [, cDatabase ]]])
```

Параметры

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

nTimeout (необязательно)

Максимальное время ожидания в секундах, пока служба находится в состоянии SERVICE_STOP_PENDING.

Если вы передадите 0, функция не будет ждать, пока служба остановится, а вместо этого немедленно вернется после отправки запроса на остановку.

Если вы опустите этот параметр или передадите NULL, тайм-аут будет установлен на тайм-аут по умолчанию, сообщенный службой. См. справку MSDN для члена "dwWaitHint" структуры [SERVICE_STATUS_PROCESS](#).

bStopDependantServices (необязательно)

по умолчанию = .F.

Если .T. службы, зависящие от переданной службы, останавливаются в первую очередь.

cServer (необязательно)

Имя сервера, на котором запущена служба.
См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.
См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

1, если служба была успешно остановлена или уже была остановлена, 0, если интервал ожидания истек до того, как служба перешла в состояние остановки.

Смотрите также

Ссылки

[ADependentServices](#)
[AServiceConfig](#)
[AServices](#)
[AServiceStatus](#)
[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[WaitForServiceStatus](#)

Используемые функции WinApi

[ControlService](#)
[EnumDependentServices](#)
[QueryServiceStatus](#)

[OpenSCManager](#)

[OpenService](#)

Str2Double

Преобразует двоичную строку в число двойной точности (64-битное числовое значение).

```
Str2Double( qValue )
```

Параметры

qValue

Двоичная строка для преобразования.

Возвращаемое значение

Числовой.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2Float

Преобразует двоичную строку в число с плавающей точкой (32-битное числовое значение).

```
Str2Float( qValue )
```

Параметры

qValue

Двоичная строка для преобразования.

Возвращаемое значение

Числовой.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2Int64

Преобразует двоичную строку в __int64 (64-битное числовое значение).

```
Str2Int64( qValue [, nFormat ])
```

Параметры

qValue

Двоичная строка для преобразования.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	Строковый литерал.
3	Двойной.

Примечание

Если вы передадите 3, значение может быть усечено!

Возвращаемое значение

Преобразованное значение в запрошенном формате.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2Long

Преобразует двоичную строку в длинное число со знаком (32-битное числовое значение).

```
Str2Long( qValue )
```

Параметры

qValue

Двоичная строка для преобразования.

Возвращаемое значение

Числовой.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2Short

Преобразует двоичную строку в короткое число со знаком (16-битное числовое значение).

```
Str2Short( qValue )
```

Параметры

qValue

Двоичная строка для преобразования.

Возвращаемое значение

Числовой.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2UInt64

Преобразует двоичную строку в беззнаковое __int64 (64-битное числовое значение).

```
Str2UInt64( qValue [, nFormat ])
```

Параметры

qValue

Двоичная строка для преобразования.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	Валюта.
2	Строковый литерал.
3	Двойной.

Примечание

Если вы передадите 3, значение может быть усечено!

Возвращаемое значение

Преобразованное значение в запрошенном формате.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2ULong

Преобразует двоичную строку в беззнаковое длинное число (32-битное числовое значение).

```
Str2ULong( qValue )
```

Параметры

qValue

Двоичная строка для преобразования.

Возвращаемое значение

Числовой.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

Str2UShort

Преобразует двоичную строку в беззнаковое короткое число (16-битное числовое значение).

```
Str2UShort( qValue )
```

Параметры

qValue

Двоичная строка для преобразования.

Возвращаемое значение

Числовой.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

StringFromCLSID

Преобразует глобальный уникальный идентификатор (GUID) в строку печатных символов.

```
StringFromCLSID( cClsId | nClsIdPointer )
```

Параметры

Возвращаемое значение

CLSID (класс GUID) в читаемой форме. Например, {2C256447-3F0D-4CBB-9D12-575BB20CDA0A}

Замечания

CLSID — это глобальный уникальный идентификатор, который идентифицирует объект класса COM. Если ваш сервер или контейнер позволяет связываться со своими встроенными объектами, вам необходимо зарегистрировать CLSID для каждого поддерживаемого класса объектов.

Пример

```
lcCLSID = CLSIDFromProgID ('VisualFoxPro.Application')  
&& lcCLSID теперь содержит CLSID в двоичном формате  
? StringFromCLSID(lcCLSID) && преобразовать в удобочитаемый  
формат
```

Смотрите также

Ссылки

[CLSIDFromProgID](#)
[CLSIDFromString](#)
[CreateGuid](#)
[CreateThreadObject](#)
[GetIUnknown](#)
[IsEqualGuid](#)
[ProgIDFromCLSID](#)
[RegisterActiveObject](#)
[RegisterObjectAsFileMoniker](#)
[RevokeActiveObject](#)

Используемые функции WinApi

[StringFromGUID2](#)

SyncToSNTPServer

Синхронизирует системное время и дату с SNTP-сервером.

```
SyncToSNTPServer( cServer [, nPort [, nTimeout ]])
```

Параметры

cServer

Имя хоста SNTP-сервера.

nPort (необязательно)

по умолчанию = 123

Порт, на котором прослушивается служба SNTP.

nTimeout (необязательно)

по умолчанию = 4000

Время ожидания в миллисекундах.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)

[AIPAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[ICmpPing](#)

[Ip2MacAddress](#)

[ResolveHostToIp](#)

[UrlDownloadToFileEx](#)

Используемые функции WinApi

[gethostbyname](#)

[GetSystemTime](#)

[socket](#)

[connect](#)

[send](#)

[recv](#)

[closesocket](#)

[SetSystemTime](#)

Timet2DT

Преобразует временную метку Time_t (Unix) в значение datetime.

```
Timet2DT( nTimestamp [, bToUTC ] )
```

Параметры

nTimestamp

Метка времени Time_t (UNIX) — числовое значение, указывающее количество секунд, прошедших с 1 января 1970 года.

bToUTC (необязательно)

по умолчанию = .F.

Если .T. возвращаемая DateTime будет в формате UTC, в противном случае — в местном часовом поясе.

Возвращаемое значение

Дата и время.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[UTC2DT](#)

UInt642Str

Преобразует беззнаковое значение __int64 (64-битное числовое значение) в требуемый формат.

```
UInt642Str( yqcnValue [, nFormat ])
```

Параметры

cnValue

Числовое значение в диапазоне от 0 до 18 446 744 073 709 551 615.

Параметр может быть передан в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "-1234567890123"
4. Как обычное числовое значение VFP.

nFormat (необязательно)

по умолчанию = 1

Указывает формат возвращаемого значения.

Одно из следующих значений.

Формат	Описание
1	8-байтовая двоичная строка.
2	Строковый литерал.

Возвращаемое значение

Беззнаковое значение __int64 в запрошенном формате.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

ULong2Str

Преобразует беззнаковое длинное (32-битное числовое значение) значение в двоичную строку.

```
ULong2Str( nValue )
```

Параметры

nValue

Числовое значение в диапазоне от 0 до 4 294 967 295.

Возвращаемое значение

Строка, которая в двоичном виде равна переданному беззнаковому длинному значению.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[UShort2Str](#)
[Value2Variant](#)
[Variant2Value](#)

UnbindEventsEx

Отменяет привязку событий для окон, которые ранее были привязаны с помощью [BindEventsEx](#).

```
UnbindEventsEx( nHwnd [, nMsg [, bClass ]])
```

Параметры

nHwnd

Указывает дескриптор окна, для которого отменяются привязки событий.

***nMsg* (необязательно)**

Сообщение, которое должно быть отвязано.

Если вы опустите параметр *nMsg* или передадите 0, все сообщения будут отвязаны, а окно будет неподклассифицировано.

***bClass* (необязательно)**

по умолчанию = .F.

Если вы создали подкласс класса окна с флагом BINDEVENTSEX_CLASSPROC, вам необходимо передать .T. в этом параметре.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[AsyncWaitForObject](#)

[BindEventsEx](#)

[CancelWaitForObject](#)

[CreateCallbackFunc](#)

[CreatePublicShadowObjReference](#)

[DestroyCallbackFunc](#)

[ReleasePublicShadowObjReference](#)

Используемые функции WinApi

[SetWindowLong](#)

[SetClassLong](#)

UnlockHGlobal

Уменьшает количество блокировок, связанных с объектом памяти, выделенным с помощью [AllocHGlobal](#).

```
UnlockHGlobal( nHandle )
```

Параметры

nHandle

Дескриптор блока памяти, возвращаемый [AllocHGlobal](#).

Возвращаемое значение

Дескриптор памяти может быть заблокирован несколько раз.

Функция возвращает 1, если дескриптор полностью разблокирован, или 2, если всё ещё есть блокировки.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[ValidateMem](#)

Используемые функции WinApi

[GlobalUnlock](#)

UrlDownloadToFileEx

Загружает ресурсы из Интернета и сохраняет их в файл.

```
UrlDownloadToFileEx( cUrl , cLocalFile [, cCallback [,
bAsynchronous ]])
```

Параметры

cUrl

Необходимо указать URL-адрес для загрузки и префикс протокола (например, http://, ftp://)!

cLocalFile

Место загрузки — допустимое локальное имя файла.

cCallback (необязательно)

Функция обратного вызова для отчета о ходе загрузки.

Функция обратного вызова должна иметь следующий прототип:

```
FUNCTION UrlProgress
LPARAMETERS ulProgress, ulProgressMax, ulStatusCode
ENDFUNC
```

значения параметров:

ulProgress = загружено байтов

ulProgressMax = размер загруженного ресурса

ulStatusCode = причина обратного вызова

ulStatusCode может быть одним из следующих:

Значение	Описание
BINDSTATUS_DOWNLOADINGDATA	Поступил новый блок данных.
BINDSTATUS_FINDINGRESOURCE	URL-адрес будет разрешен.
BINDSTATUS_CONNECTING	Подключение.
BINDSTATUS_REDIRECTING	Перенаправление.
BINDSTATUS_BEGINDOWNLOADDATA	Начинается загрузка.
BINDSTATUS_ENDDOWNLOADDATA	Загрузка завершена, файл теперь находится в локальном кэше IE.
BINDSTATUS_DOWNLOAD_FINISHED	Теперь загруженный файл готов к чтению из указанного места назначения.
BINDSTATUS_DOWNLOAD_ABORTED	Загрузка была прервана (через AbortUrlDownloadToFileEx или освобождение FLL).
BINDSTATUS_USINGCACHEDCOPY	Файл находится в кэше IE и не загружается повторно.
BINDSTATUS_SENDINGREQUEST	Файл запрошен.

bAsynchronous (необязательно)

по умолчанию = .F.

Если передано .T., загрузка выполняется асинхронно в отдельном потоке, функция возвращается немедленно. В противном случае загрузка выполняется синхронно, и функция возвращается после завершения загрузки.

Обратный вызов Progress работает для обоих типов.

Чтобы прервать асинхронную загрузку, необходимо вызвать [AbortUrlDownloadToFileEx](#) с числовым значением дескриптора, возвращенным UrlDownloadToFileEx.

Чтобы прервать синхронную загрузку, необходимо вернуть .F. из функции обратного вызова.

Возвращаемое значение

Если загрузка синхронная, функция возвращает результат из API UrlDownloadToFile (HRESULT | < 0 неудача | > 0 успех).

Если загрузка асинхронная, функция возвращает внутренний дескриптор, представляющий поток, который выполняет загрузку.

Смотрите также

Ссылки

[AbortUrlDownloadToFileEx](#)

[AIPAddresses](#)

[ANetFiles](#)

[ANetServers](#)

[GetServerTime](#)

[ICmpPing](#)

[Ip2MacAddress](#)

[Resolve HostToIp](#)

[SyncToSNTPServer](#)

Используемые функции WinAPI

[URLDownloadToFile](#)

UShort2Str

Преобразует беззнаковое короткое (16-битное числовое значение) значение в двоичную строку.

```
UShort2Str( nValue )
```

Параметры

nValue

Числовое значение в диапазоне от 0 до 65 535.

Возвращаемое значение

Строка, которая в двоичном виде равна переданному короткому значению без знака.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[Value2Variant](#)
[Variant2Value](#)

UTC2DT

Преобразует значение даты и времени из UTC в местный часовой пояс.

```
UTC2DT( tTime )
```

Параметры

tTime

Значение даты и времени для преобразования.

Возвращаемое значение

Дата и время.

Смотрите также

Ссылки

[ATimeZones](#)

[Double2DT](#)

[DT2Double](#)

[DT2FT](#)

[DT2ST](#)

[DT2Timet](#)

[DT2UTC](#)

[FT2DT](#)

[GetSystemTimeEx](#)

[SetSystemTimeEx](#)

[ST2DT](#)

[Timet2DT](#)

ValidateMem

Проверяет внутреннюю кучу библиотеки. Функция сканирует все блоки памяти в куче и проверяет, что структуры управления кучей, поддерживаемые менеджером кучи, находятся в согласованном состоянии. Вы также можете использовать функцию ValidateMem для проверки одного блока памяти без проверки действительности всей кучи.

```
ValidateMem( nAddress )
```

Параметры

nAddress

Указатель на блок памяти, возвращаемый [AllocMem](#). Этот параметр может быть равен 0.

Если этот параметр равен 0, функция пытается проверить всю кучу.

Если этот параметр не равен 0, функция пытается проверить блок памяти, на который указывает *nAddress*.

Она не пытается проверить остальную часть кучи.

Возвращаемое значение

Возвращает .T., если указанная куча или блок памяти допустимы, в противном случае — .F.

Замечания

Когда вы используете ValidateMem для проверки одного блока памяти в куче, он проверяет только структуры управления, относящиеся к этому элементу. [HeapValidate](#) может проверять только выделенные блоки памяти. Вызов ValidateMem для освобожденного блока памяти вернет .F., поскольку нет структур управления для проверки.

Смотрите также

Ссылки

[AllocHGlobal](#)
[AllocMem](#)
[AllocMemTo](#)
[AMemBlocks](#)
[CompactMem](#)
[FreeHGlobal](#)
[FreeMem](#)
[FreePMem](#)
[FreeRefArray](#)
[LockHGlobal](#)
[ReAllocHGlobal](#)
[ReAllocMem](#)
[SizeOfMem](#)
[UnlockHGlobal](#)

Используемые функции WinApi

[HeapValidate](#)

Value2Variant

Преобразует переменную или поле любого типа в двоичную строку.

```
Value2Variant( cValue )
```

Параметры

cValue

Допустим любой тип данных (включая NULL).

Возвращается оптимизированная по размеру структура LCK Value и фактические данные для типов символов, которые можно сохранить в поле MEMO NOCPTRANS или BLOB и затем десериализовать в исходное значение с помощью [Variant2Value](#).

Совет

[SET BLOCKSIZE TO 0](#) — для таблицы, чтобы сделать хранение более эффективным.

Возвращаемое значение

Строка.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Variant2Value](#)

ValueToSQLVariant

Преобразует значение FoxPro в двоичное значение для хранения в двоичном значении SQL_SS_VARIANT, для хранения внутри SQL Server.

```
ValueToSQLVariant( cValue [, nSQLType [, nScale ]])
```

Параметры

cValue

Любое значение FoxPro, допустимое как SQL_SS_VARIANT, более или менее все типы FoxPro размером менее 8000 байт.

nSQLType (необязательно)

По умолчанию соответствующий тип выводится из переданного типа FoxPro. Но если вы извлекли какое-то значение в двоичной форме (например, SQL_SS_TIME) и хотите сохранить его снова, вам придется явно передать тип.

nScale (необязательно)

Требуется только для хранения определенных двоичных типов, таких как SQL_SS_TIME, определяет количество дробных цифр.

Возвращаемое значение

Смотрите также

Ссылки

- [ASQLDataSources](#)
- [ASQLDrivers](#)
- [ChangeSQLDataSource](#)
- [CreateSQLDataSource](#)
- [DeleteSQLDataSource](#)
- [SQLCancelEx](#)
- [SQLExecEx](#)
- [SQLGetPropEx](#)
- [SQLPrepareEx](#)
- [SQLSetPropEx](#)
- [SQLRegisterTableType](#)
- [SQLCreateTableTypeCursor](#)
- [SQLTablesEx](#)
- [SQLColumnsEx](#)
- [SQLPrimaryKeysEx](#)
- [SQLForeignKeysEx](#)
- [SQLProceduresEx](#)
- [SQLProcedureColumnsEx](#)
- [SQLTablePrivilegesEx](#)
- [SQLColumnPrivilegesEx](#)
- [SQLSpecialColumnsEx](#)
- [SQLVariantToValue](#)

Variant2Value

Преобразует двоичную строку, созданную [Value2Variant](#), в исходное значение.

```
Variant2Value( variant )
```

Параметры

variant

Двоичная строка, созданная [Value2Variant](#).

Примечание

Поля MEMO NOCPTRANS и BLOB должны передаваться по ссылке в эту функцию!

Возвращаемое значение

Универсальный тип данных.

Смотрите также

Ссылки

[Colors2RGB](#)
[Double2Str](#)
[Float2Str](#)
[Int642Str](#)
[Long2Str](#)
[Num2Binary](#)
[PG_ByteA2Str](#)
[PG_Str2ByteA](#)
[RGB2Colors](#)
[Short2Str](#)
[Str2Double](#)
[Str2Float](#)
[Str2Int64](#)
[Str2Long](#)
[Str2Short](#)
[Str2UInt64](#)
[Str2ULong](#)
[Str2UShort](#)
[UInt642Str](#)
[ULong2Str](#)
[UShort2Str](#)
[Value2Variant](#)

VFP2C32

Фиктивная функция для проверки того, загружена ли библиотека.

VFP2C32 ()

Возвращаемое значение

Возвращает .F.

Замечания

Если библиотека не загружена, вызывается ваша функция FoxPro. После загрузки библиотеки функция FLL имеет приоритет и вызывается вместо нее.

Пример

```
FUNCTION VFP2C32 ()
SET LIBRARY TO LOCFIL('vfp2c32.fll')
RETURN .T.
ENDFUNC
```

VFP2CSys

Предоставляет доступ к внутренним ресурсам библиотеки или изменяет глобальное поведение библиотеки.

```
VFP2CSys( nOption [, nNewValue ])
```

Параметры

nOption

Option	Возвращаемое значение
1	HMODULE vfp2c32.fll
2	HANDLE к внутренней куче библиотеки, используемой для маршалинга
3	кодировка страницы по умолчанию, используемая для преобразования Ansi - Unicode - по умолчанию CP_ACP
4	автоматическое преобразование Unicode в Ansi в SqlExecEx — по умолчанию .T.
5	сохранять информацию из функций ODBC, которые возвращают SQL_SUCCESS_WITH_INFO внутри SQLExecEx в структуре ошибок — по умолчанию .F. — это предназначено для отладки
6	сопоставить двоичные поля внутри SQLExecEx с общими полями (тип данных G) — по умолчанию .F.

nNewValue (необязательно)

Следующие параметры могут быть изменены.

Option	Описание
3	nNewValue должно быть новым значением кодировки страницы, которое используется во всех преобразованиях строк Ansi в Unicode.
4	nNewValue должно быть .T. или .F.
5	nNewValue должно быть .T. или .F.
6	nNewValue должно быть .T. или .F.

Возвращаемое значение

Возвращаемое значение зависит от переданного значения.

Смотрите также

Ссылки

[AErrorEx](#)

[FormatMessageEx](#)

WaitForServiceStatus

Отслеживает достижение службой Windows указанного состояния.

```
WaitForServiceStatus( cServiceName | nServiceHandle , nStatus [,  
nTimeout [, cServer [, cDatabase ]]])
```

Параметры

cServiceName | nServiceHandle

Либо имя службы, либо числовой дескриптор, возвращаемый функцией [OpenServiceEx](#).

nStatus

Статус обслуживания, которого следует ожидать.

Одно из следующих значений.

Статус	Описание
SERVICE_STOPPED	Служба не запущена.
SERVICE_START_PENDING	Служба начинается.
SERVICE_STOP_PENDING	Служба останавливается.
SERVICE_RUNNING	Услуга запущена.
SERVICE_CONTINUE_PENDING	Ожидается продолжение обслуживания.
SERVICE_PAUSE_PENDING	Ожидается приостановка обслуживания.
SERVICE_PAUSED	Предоставление услуги приостановлено.

nTimeout (необязательно)

Максимальное время ожидания в секундах, пока служба не находится в запрошенном состоянии.

Если вы передадите 0 в качестве значения таймаута, функция проверит статус один раз и затем немедленно вернет управление.

Если вы опустите этот параметр или передадите NULL, таймаут будет установлен на значение таймаута по умолчанию, сообщенное службой.

См. справку MSDN для члена "dwWaitHint" структуры [SERVICE_STATUS_PROCESS](#).

cServer (необязательно)

Имя сервера, на котором запущена служба.

См. справку MSDN для [OpenSCManager](#).

cDatabase (необязательно)

База данных, в которой зарегистрирована служба.

См. справку MSDN для [OpenSCManager](#).

Возвращаемое значение

1, если служба находится в запрошенном состоянии в течение периода ожидания, 0 — в противном случае.

Смотрите также

Ссылки

[ADependentServices](#)

[AServiceConfig](#)

[AServices](#)

[AServiceStatus](#)

[CloseServiceHandleEx](#)
[ContinueService](#)
[ControlServiceEx](#)
[CreateServiceEx](#)
[OpenServiceEx](#)
[PauseService](#)
[StartServiceEx](#)
[StopServiceEx](#)

Используемые функции WinApi

[QueryServiceStatus](#)
[OpenSCManager](#)
[OpenService](#)

WriteBytes

Записывает двоичные данные по указанному адресу памяти.

```
WriteBytes( nAddress , cValue [, nMaxBytes ])
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

cValue

Данные для записи.

nMaxBytes (необязательно)

по умолчанию = [LEN](#)(cValue)

Максимальное количество байтов для записи.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteChar](#)
- [WriteCharArray](#)

[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[memcpy](#)

WriteChar

Записывает один символ по указанному адресу памяти.

```
WriteChar( nAddress , cValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

cValue

Один символ.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)

[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteCharArray

Записывает строку по указанному адресу памяти.

```
WriteCharArray( nAddress , cValue [, nMaxChars ] )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

cValue

Строка для записи.

***nMaxChars* (необязательно)**

по умолчанию = [LEN](#)(*cValue*)

Максимальное количество символов для записи.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)

[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteCString

Выделяет или перераспределяет строку в стиле C.

```
WriteCString( nAddress , cValue )
```

Параметры

nAddress

Либо 0, либо ранее выделенный указатель.

Если 0, выделяется новая область памяти, достаточно большая, чтобы вместить строку в cValue + 1 для nullterminator, и строка копируется в нее.

Если nAddress не равен 0, то предполагается, что это указатель на область памяти, ранее выделенную WriteCString или [AllocMem](#).

cValue

Строка для выделения.

Возвращаемое значение

Указатель на выделенную строку C.

Замечания

Указатель, возвращаемый этой функцией, необходимо освободить с помощью [FreeMem](#), в противном случае возникнет утечка памяти.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)

[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[HeapAlloc](#)
[HeapReAlloc](#)
[memcpy](#)

WriteDouble

Записывает число двойной точности (64-битное с плавающей точкой) по указанному адресу.

```
WriteDouble( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне 1,7E +/- 308 (15 цифр).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteFloat](#)
- [WriteGPCString](#)
- [WriteInt](#)

[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteFloat

Записывает число с плавающей точкой (32-битное число с плавающей точкой) по указанному адресу.

```
WriteFloat( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне 3,4E +/- 38 (7 цифр).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteGPCString

Выделяет или перевыделяет память для строки в стиле С и записывает указатель на эту строку по указанному адресу.

```
WriteGPCString( nAddress , cValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

cValue

Строка для записи или NULL, если должен быть записан нулевой указатель.

Возвращаемое значение

Указатель на выделенную строку С или 0, если был записан NULL.

Замечания

Указатель, возвращаемый этой функцией, необходимо освободить с помощью [FreeHGlobal](#), в противном случае возникнет утечка памяти.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)

[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[GlobalAlloc](#)
[GlobalReAlloc](#)
[GlobalFree](#)

WriteInt

Записывает 32-битное целое число по указанному адресу.

```
WriteInt( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от -2 147 483 648 до 2 147 483 647.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteInt64

Записывает 64-битное целое число со знаком по указанному адресу.

```
WriteInt64( nAddress , yqcnValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

yqcnValue

Числовое значение в диапазоне от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807.

Параметр может быть передан в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "-1234567890123" .
4. Как обычное числовое значение VFP.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)

[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteInt8

Записывает 8-битное целое число по указанному адресу.

```
WriteInt8( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от -128 до 127.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteLogical

Записывает логическое значение по указанному адресу.

```
WriteLogical( nAddress , bValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

bValue

Логическое значение - .T. или .F.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUSHort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePChar

Записывает один символ по указанному косвенному адресу.

```
WritePChar( nAddress , cValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

cValue

Один символ.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePCString

Выделяет или перевыделяет память для строки в стиле С и записывает указатель на эту строку по указанному адресу.

```
WritePCString( nAddress , cValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

cValue

Строка для записи или NULL, если должен быть записан нулевой указатель.

Возвращаемое значение

Указатель на выделенную строку С или 0, если был записан NULL.

Замечания

Указатель, возвращаемый этой функцией, необходимо освободить с помощью [FreePMem](#), в противном случае возникнет утечка памяти.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)

[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[HeapAlloc](#)
[HeapReAlloc](#)
[HeapFree](#)

WritePDouble

Записывает число двойной точности (64-битное с плавающей точкой) по указанному косвенному адресу.

```
WritePDouble( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне 1,7E +/- 308 (15 цифр).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)

[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePFloat

Записывает число с плавающей точкой (32-битное число с плавающей точкой) по указанному косвенному адресу.

```
WritePFloat( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне 3,4E +/- 38 (7 цифр).

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)

[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePint

Записывает 32-битное целое число по указанному косвенному адресу.

```
WritePint( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от -2 147 483 648 до 2 147 483 647.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUSHort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePInt64

Записывает 64-битное целое число со знаком по указанному косвенному адресу.

```
WritePInt64( nAddress , yqcnValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

yqcnValue

Числовое значение в диапазоне от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807.

Параметр может быть передан в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "-1234567890123" .
4. Как обычное числовое значение VFP.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)

[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePInt8

Записывает 8-битное целое число по указанному косвенному адресу.

```
WritePInt8( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от -128 до 127.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePLogical

Записывает логическое значение по указанному косвенному адресу.

```
WritePLogical( nAddress , bValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

bValue

Логическое значение - .T. или .F.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePointer

Записывает указатель по указанному адресу.

```
WritePointer( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUSHort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)

[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePPointer

Записывает указатель по указанному косвенному адресу.

```
WritePPointer( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение указателя.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUSHort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePShort

Записывает 16-битное целое число по указанному косвенному адресу.

```
WritePShort( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от -32 768 до 32 767.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePUInt

Записывает 32-битное беззнаковое целое число по указанному косвенному адресу.

```
WritePUInt( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от 0 до 4 294 967 295.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUSHort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePUInt64

Записывает 64-битное беззнаковое целое число по указанному косвенному адресу.

```
WritePUInt64( nAddress , yqcnValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

yqcnValue

Числовое значение в диапазоне от 0 до 9 223 372 036 854 775 807.

Параметр может быть передан в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "1234567890123" .
4. Как обычное числовое значение VFP.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)

[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePUInt8

Записывает 8-битное беззнаковое целое число по указанному косвенному адресу.

```
WritePUInt8( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от 0 до 255.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUSHort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePUSHort

Записывает 16-битное беззнаковое целое число по указанному косвенному адресу.

```
WritePUSHort( nAddress , nValue )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от 0 до 65 535.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WritePWChar

Записывает один символ Unicode по указанному косвенному адресу.

```
WritePWChar( nAddress , cValue [, nCodePage ] )
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

cValue

Один символ.

***nCodePage* (необязательно)**

по умолчанию = кодовая страница, заданная с помощью [VFP2CSys](#)(4, *nCodePage*), которая по умолчанию равна CP_ACP

Кодовая страница, используемая при выполнении преобразования ANSI в Unicode. Этот параметр может быть установлен на значение любой кодовой страницы, установленной или доступной в операционной системе.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)

[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[MultiByteToWideChar](#)

WritePWString

Выделяет или перераспределяет память для строки Unicode в стиле C и записывает указатель на эту строку по указанному адресу.

```
WritePWString( nAddress , cValue [, nCodePage ])
```

Параметры

nAddress

Косвенный адрес памяти, по которому должны быть записаны данные.

cValue

Строка для записи или NULL, если должен быть записан нулевой указатель.

***nCodePage* (необязательно)**

по умолчанию = кодовая страница, заданная с помощью [VFP2CSys](#)(4, *nCodePage*), которая по умолчанию равна CP_ACP

Кодовая страница, используемая при выполнении преобразования ANSI в Unicode. Этот параметр может быть установлен на значение любой кодовой страницы, установленной или доступной в операционной системе.

Возвращаемое значение

Указатель на выделенную строку Unicode C или 0, если был записан NULL.

Замечания

Указатель, возвращаемый этой функцией, необходимо освободить с помощью [FreePMem](#), в противном случае возникнет утечка памяти.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)

[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[MultiByteToWideChar](#)

WriteRegistryKey

Устанавливает данные и тип указанного значения в разделе реестра.

```
WriteRegistryKey( nRegKey , vValue [, cValueName [, cKeyName [,
nValueType ]]])
```

Параметры

nRegKey

Либо дескриптор реестра, возвращенный [OpenRegistryKey](#), либо одна из следующих констант ключа.

Константа	Описание
HKEY_CLASSES_ROOT	<p>Записи реестра, подчиненные этому ключу, определяют типы (или классы) документов и свойства, связанные с этими типами. Приложения Shell и COM используют информацию, хранящуюся в этом ключе.</p> <p>Этот ключ также обеспечивает обратную совместимость с регистрационной базой данных Windows 3.1, сохраняя информацию для поддержки DDE и OLE. Просмотрщики файлов и расширения пользовательского интерфейса хранят свои идентификаторы классов OLE в HKEY_CLASSES_ROOT, а внутрипроцессные серверы регистрируются в этом ключе.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей.</p>
HKEY_CURRENT_CONFIG	<p>Содержит информацию о текущем профиле оборудования локальной компьютерной системы. Информация в разделе HKEY_CURRENT_CONFIG описывает только различия между текущей конфигурацией оборудования и стандартной конфигурацией. Информация о стандартной конфигурации оборудования хранится в разделах Software и System раздела HKEY_LOCAL_MACHINE.</p> <p>HKEY_CURRENT_CONFIG — это псевдоним для HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current.</p>
HKEY_CURRENT_USER	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя. Эти предпочтения включают настройки переменных среды, данные о группах программ, цветах, принтерах, сетевых подключениях и предпочтениях приложений. Этот ключ упрощает установку настроек текущего пользователя; ключ сопоставляется с ветвью текущего пользователя в HKEY_USERS. В HKEY_CURRENT_USER поставщики программного обеспечения хранят текущие пользовательские предпочтения, которые будут использоваться в их приложениях. Например, Microsoft создает ключ HKEY_CURRENT_USER\Software\Microsoft для использования своими приложениями, причем каждое приложение создает свой собственный подраздел в ключе Microsoft.</p> <p>Сопоставление между HKEY_CURRENT_USER и HKEY_USERS выполняется для каждого процесса и</p>

	<p>устанавливается при первой ссылке процесса на HKEY_CURRENT_USER. Сопоставление основано на контексте безопасности первого потока, ссылающегося на HKEY_CURRENT_USER. Если этот контекст безопасности не имеет куста реестра, загруженного в HKEY_USERS, сопоставление устанавливается с помощью HKEY_USERS\.Default. После того, как это сопоставление установлено, оно сохраняется, даже если контекст безопасности потока изменяется.</p> <p>Все записи реестра в HKEY_CURRENT_USER, за исключением тех, что находятся в HKEY_CURRENT_USER\Software\Classes, включены в часть реестра для каждого пользователя перемещаемого профиля пользователя. Чтобы исключить другие записи из перемещаемого профиля пользователя, сохраните их в HKEY_CURRENT_USER_LOCAL_SETTINGS.</p> <p>Этот дескриптор не должен использоваться в службе или приложении, которые выдают себя за разных пользователей. Вместо этого вызовите функцию RegOpenCurrentUser.</p>
HKEY_CURRENT_USER_LOCAL_SETTINGS	<p>Записи реестра, подчиненные этому ключу, определяют предпочтения текущего пользователя, которые являются локальными для машины. Эти записи не включены в часть реестра пользователя перемещаемого профиля пользователя.</p> <p>Windows Server 2008, Windows Vista, Windows Server 2003 и Windows XP/2000: этот ключ поддерживается, начиная с Windows 7 и Windows Server 2008 R2.</p>
HKEY_LOCAL_MACHINE	<p>Записи реестра, подчиненные этому ключу, определяют физическое состояние компьютера, включая данные о типе шины, системной памяти и установленном оборудовании и программном обеспечении. Он содержит подразделы, которые содержат текущие данные конфигурации, включая информацию Plug and Play (ветвь Enum, которая включает полный список всего оборудования, которое когда-либо было в системе), настройки входа в сеть, информацию о безопасности сети, информацию, связанную с программным обеспечением (такую как имена серверов и местоположение сервера), и другую системную информацию.</p>
HKEY_PERFORMANCE_DATA	<p>Записи реестра, подчиненные этому ключу, позволяют получить доступ к данным о производительности. Данные фактически не хранятся в реестре; функции реестра заставляют систему собирать данные из их источника.</p>
HKEY_PERFORMANCE_NLSTEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются на текстовые строки, описывающие счетчики на локальном языке области, в которой работает компьютерная система. Эти записи недоступны для Regedit.exe и Regedt32.exe.</p> <p>Windows 2000: этот ключ не поддерживается.</p>
HKEY_PERFORMANCE_TEXT	<p>Записи реестра, подчиненные этому ключу, ссылаются</p>

	на текстовые строки, описывающие счетчики на американском английском. Эти записи недоступны для Regedit.exe и Regedt32.exe.
	Windows 2000: этот ключ не поддерживается.
HKEY_USERS	Записи реестра, подчиненные этому ключу, определяют конфигурацию пользователя по умолчанию для новых пользователей на локальном компьютере и конфигурацию пользователя для текущего пользователя.

vValue

Значение для записи в реестр.
Может быть любого типа, кроме NULL.

cValueName (необязательно)

Имя_значения, под которым должно храниться значение.
Если вы опустите cValueName или передадите пустую строку, значение будет сохранено в значении по умолчанию указанного ключа.

sKeyName (необязательно)

Путь к подключаемому ключу ключа, переданного в nRegKey.

Например

```
&& запись в HKEY_LOCAL_MACHINE\SOFTWARE\YourFirm\YourApp\Settings
WriteRegistryKey(HKEY_LOCAL_MACHINE, 'SomeOption', ;
    'NameOfOption', 'SOFTWARE\YourFirm\YourApp\Settings')
```

nValueType (необязательно)

Тип данных значения реестра.

Если тип данных не указан, он определяется типом параметра vValue .
В следующей таблице показано сопоставление VFP с типами реестра по умолчанию.

Тип FoxPro	Тип реестра
C	REG_SZ
C binary **	REG_BINARY
I	REG_INTEGER ***
N	REG_DOUBLE ***
D	REG_DATE ***
T	REG_DATETIME ***
L	REG_LOGICAL ***
Y	REG_MONEY ***

Примечание

** Из поля C NOCPTRANS, M NOCPTRANS, Q или W или из выражения, созданного с помощью [CREATEBINARY](#) ().

Примечание

*** Эти типы определяются пользователем. Если вы всегда опускаете параметр nValueType при вызове [ReadRegistryKey](#) и WriteRegistryKey, вы всегда будете получать точное значение VFP.

Если вы укажете тип самостоятельно, эти типы будут действительны для указанного типа данных FoxPro.

Тип FoxPro	Тип реестра
C, Q	All
I	REG_DWORD, REG_QWORD, REG_DOUBLE, REG_INTEGER, REG_BINARY
N	REG_DWORD, REG_QWORD, REG_DOUBLE, REG_INTEGER, REG_BINARY
D	REG_DATE, REG_DOUBLE, REG_BINARY
T	REG_DATETIME, REG_DOUBLE, REG_BINARY
L	REG_LOGICAL, REG_DWORD, REG_INTEGER, REG_BINARY
Y	REG_MONEY, REG_QWORD, REG_BINARY

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ARegistryKeys](#)
[ARegistryValues](#)
[CancelRegistryChange](#)
[CloseRegistryKey](#)
[CreateRegistryKey](#)
[DeleteRegistryKey](#)
[FindRegistryChange](#)
[OpenRegistryKey](#)
[ReadRegistryKey](#)
[RegistryHiveToObject](#)
[RegistryValuesToObject](#)

Используемые функции WinApi

[RegSetValueEx](#)
[RegOpenKeyEx](#)
[RegCloseKey](#)

WriteShort

Записывает 16-битное целое число по указанному адресу.

```
WriteShort( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от -32 768 до 32 767.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteUInt

Записывает 32-битное беззнаковое целое число по указанному адресу.

```
WriteUInt( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от 0 до 4 294 967 295.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteUInt64

Записывает 64-битное беззнаковое целое число по указанному адресу.

```
WriteUInt64( nAddress , yqcnValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

qcnValue

Числовое значение в диапазоне от 0 до 18 446 744 073 709 551 615.

Параметр может быть передан в 4 различных форматах.

1. Как значение валюты.
2. Как 8-байтовая строка varbinary.
3. Как строковый литерал, например "1234567890123".
4. Как обычное числовое значение VFP.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)

[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteUInt8

Записывает 8-битное беззнаковое целое число по указанному адресу.

```
WriteUInt8( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от 0 до 255.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteUShort

Записывает 16-битное беззнаковое целое число по указанному адресу.

```
WriteUShort( nAddress , nValue )
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

nValue

Числовое значение в диапазоне от 0 до 65 535.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUShort](#)
- [ReadPWString](#)
- [ReadShort](#)
- [ReadUInt](#)
- [ReadUInt64](#)
- [ReadUInt8](#)
- [ReadUShort](#)
- [ReadWCharArray](#)
- [ReadWString](#)
- [WriteBytes](#)
- [WriteChar](#)
- [WriteCharArray](#)
- [WriteCString](#)
- [WriteDouble](#)
- [WriteFloat](#)
- [WriteGPCString](#)

[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteWChar](#)
[WriteWCharArray](#)
[WriteWString](#)

WriteWChar

Записывает один символ Unicode по указанному адресу.

```
WriteWChar( nAddress , cChar [, nCodePage ])
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

cChar

Один символ.

***nCodePage* (необязательно)**

по умолчанию = кодовая страница, заданная с помощью [VFP2CSys](#)(4, *nCodePage*), которая по умолчанию равна CP_ACP

Кодовая страница, используемая при выполнении преобразования ANSI в Unicode. Этот параметр может быть установлен на значение любой кодовой страницы, установленной или доступной в операционной системе.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)
[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)

[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWCharArray](#)
[WriteWString](#)

Используемые функции WinApi

[MultiByteToWideChar](#)

WriteWCharArray

Записывает строку Unicode по указанному адресу.

```
WriteWCharArray( nAddress , cValue [, nMaxChars [, nCodePage ]])
```

Параметры

nAddress

Адрес памяти, по которому должны быть записаны данные.

cValue

Строка для записи.

***nMaxChars* (необязательно)**

по умолчанию = [LEN](#)(cValue)

Максимальное количество символов для записи.

***nCodePage* (необязательно)**

по умолчанию = кодовая страница, заданная с помощью [VFP2CSys](#)(4, nCodePage), которая по умолчанию равна CP_ACP

Кодовая страница, используемая при выполнении преобразования ANSI в Unicode. Этот параметр может быть установлен на значение любой кодовой страницы, установленной или доступной в операционной системе.

Возвращаемое значение

Всегда .T.

Смотрите также

Ссылки

- [ReadBytes](#)
- [ReadChar](#)
- [ReadCharArray](#)
- [ReadCString](#)
- [ReadDouble](#)
- [ReadFloat](#)
- [ReadInt](#)
- [ReadInt64](#)
- [ReadInt8](#)
- [ReadLogical](#)
- [ReadPChar](#)
- [ReadPCString](#)
- [ReadPDouble](#)
- [ReadPFloat](#)
- [ReadPInt](#)
- [ReadPInt64](#)
- [ReadPInt8](#)
- [ReadPLogical](#)
- [ReadPointer](#)
- [ReadPPointer](#)
- [ReadProcessMemoryEx](#)
- [ReadPShort](#)
- [ReadPUInt](#)
- [ReadPUInt64](#)
- [ReadPUInt8](#)
- [ReadPUSHort](#)
- [ReadPWString](#)
- [ReadShort](#)

[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUSHort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWString](#)

Используемые функции WinApi

[MultiByteToWideChar](#)

WriteWString

Выделяет или перераспределяет строку Unicode в стиле C.

```
WriteWString( nAddress , cValue [, nCodePage ] )
```

Параметры

nAddress

Либо 0, либо ранее выделенный указатель.

Если выделяется 0, новая область памяти, достаточно большая, чтобы вместить строку в **cValue** + 1 для nullterminator, и строка копируется.

Если nAddress не равен 0, предполагается, что это указатель на область памяти, ранее выделенную WriteWString или [AllocMem](#).

cValue

Строка для записи или NULL, если должен быть записан нулевой указатель.

nCodePage (необязательно)

по умолчанию = кодовая страница, заданная с помощью [VFP2CSys](#)(4, nCodePage), которая по умолчанию равна CP_ACP

Кодовая страница, используемая при выполнении преобразования ANSI в Unicode. Этот параметр может быть установлен на значение любой кодовой страницы, установленной или доступной в операционной системе.

Возвращаемое значение

Указатель на выделенную строку.

Замечания

Указатель, возвращаемый этой функцией, необходимо освободить с помощью [FreeMem](#), в противном случае возникнет утечка памяти.

Смотрите также

Ссылки

[ReadBytes](#)
[ReadChar](#)
[ReadCharArray](#)
[ReadCString](#)
[ReadDouble](#)
[ReadFloat](#)
[ReadInt](#)
[ReadInt64](#)
[ReadInt8](#)
[ReadLogical](#)
[ReadPChar](#)
[ReadPCString](#)
[ReadPDouble](#)
[ReadPFloat](#)
[ReadPInt](#)
[ReadPInt64](#)
[ReadPInt8](#)
[ReadPLogical](#)
[ReadPointer](#)
[ReadPPointer](#)
[ReadProcessMemoryEx](#)
[ReadPShort](#)
[ReadPUInt](#)
[ReadPUInt64](#)

[ReadPUInt8](#)
[ReadPUShort](#)
[ReadPWString](#)
[ReadShort](#)
[ReadUInt](#)
[ReadUInt64](#)
[ReadUInt8](#)
[ReadUShort](#)
[ReadWCharArray](#)
[ReadWString](#)
[WriteBytes](#)
[WriteChar](#)
[WriteCharArray](#)
[WriteCString](#)
[WriteDouble](#)
[WriteFloat](#)
[WriteGPCString](#)
[WriteInt](#)
[WriteInt64](#)
[WriteInt8](#)
[WriteLogical](#)
[WritePChar](#)
[WritePCString](#)
[WritePDouble](#)
[WritePFloat](#)
[WritePInt](#)
[WritePInt64](#)
[WritePInt8](#)
[WritePLogical](#)
[WritePointer](#)
[WritePPointer](#)
[WritePShort](#)
[WritePUInt](#)
[WritePUInt64](#)
[WritePUInt8](#)
[WritePUShort](#)
[WritePWChar](#)
[WritePWString](#)
[WriteShort](#)
[WriteUInt](#)
[WriteUInt64](#)
[WriteUInt8](#)
[WriteUShort](#)
[WriteWChar](#)
[WriteWCharArray](#)

Используемые функции WinApi

[MultiByteToWideChar](#)