

Ввод с помощью клавиатуры и мыши

Статья 11.03.2023

Общие сведения о технологии ввода с помощью клавиатуры и мыши.

Технология ввода с помощью клавиатуры и мыши не связана с заголовками.

Рекомендации по программированию для этой технологии см. в следующих разделах:

- [Ввод с помощью клавиатуры и мыши](#)

Перечисления

`TOOLTIP_DISMISS_FLAGS`

Функции

[_TrackMouseEvent](#)

Публикует сообщения, когда указатель мыши покидает окно или наносит указатель мыши на окно в течение указанного периода времени. Эта функция вызывает `TrackMouseEvent`, если она существует, в противном случае эмулирует ее.

[ActivateKeyboardLayout](#)

Задает идентификатор входного языкового стандарта (прежнее название — дескриптор раскладки клавиатуры) для вызывающего потока или текущего процесса. Идентификатор входного языкового стандарта указывает языковой стандарт, а также физическую раскладку клавиатуры.

[BlockInput](#)

Блокирует доступ к приложениям событий ввода с помощью клавиатуры и мыши.

[DefRawInputProc](#)

Проверяет правильность размера структуры `RAWINPUTHEADER`.

[DragDetect](#)

Захватывает мышь и отслеживает ее движение, пока пользователь не отпустит левую кнопку мыши, не нажмет клавишу ESC или не переместит мышь за пределы прямоугольника перетаскивания, в котором находится указанная точка.

[EnableWindow](#)

Включает или отключает ввод с помощью мыши и клавиатуры для указанного окна или элемента управления. Если входные данные отключены, окно не получает такие входные данные, как щелчки мышью и нажатия клавиш. Если входные данные включены, окно получает все входные данные.

[GET_APPCOMMAND_LPARAM](#)

Извлекает команду приложения из указанного значения LPARAM.

[GET_DEVICE_LPARAM](#)

Извлекает тип устройства ввода из указанного значения LPARAM.

[GET_FLAGS_LPARAM](#)

Извлекает состояние определенных виртуальных ключей из указанного значения LPARAM.
([GET_FLAGS_LPARAM](#))

[GET_KEYSTATE_LPARAM](#)

Извлекает состояние определенных виртуальных ключей из указанного значения LPARAM.
([GET_KEYSTATE_LPARAM](#))

[GET_KEYSTATE_WPARAM](#)

Извлекает состояние определенных виртуальных ключей из указанного значения WPARAM.

[GET_NCHITTEST_WPARAM](#)

Извлекает значение проверки попадания из указанного значения WPARAM.

[GET_RAWINPUT_CODE_WPARAM](#)

Извлекает входной код из wParam в WM_INPUT.

[GET_WHEEL_DELTA_WPARAM](#)

Извлекает значение wheel-delta из указанного значения WPARAM.

[GET_XBUTTON_WPARAM](#)

Извлекает состояние определенных кнопок из указанного значения WPARAM.

[GetActiveWindow](#)

Извлекает дескриптор окна к активному окну, подключенному к очереди сообщений вызывающего потока.

[GetAsyncKeyState](#)

Определяет, находится ли клавиша вверх или вниз во время вызова функции и была ли клавиша нажата после предыдущего вызова GetAsyncKeyState.

[GetCapture](#)

Извлекает дескриптор в окно (если таковое имеется), захватив мышь. Только одно окно за раз может захватывать мышь; Это окно получает ввод мыши независимо от того, находится ли курсор в его границах.

[GetDoubleClickTime](#)

Извлекает текущее время двойного щелчка мыши.

[GetFocus](#)

Извлекает дескриптор для окна с фокусом клавиатуры, если окно подключено к очереди сообщений вызывающего потока.

[GetKBCodePage](#)

Извлекает текущую кодовую страницу.

[GetKeyboardLayout](#)

Извлекает идентификатор активного языкового стандарта ввода (ранее название — раскладка клавиатуры).

[GetKeyboardLayoutList](#)

Извлекает идентификаторы входных языковых стандартов (ранее называемые дескрипторами раскладки клавиатуры), соответствующие текущему набору языковых стандартов ввода в системе. Функция копирует идентификаторы в указанный буфер.

[GetKeyboardLayoutNameA](#)

Извлекает имя идентификатора активного языкового стандарта ввода (ранее называвшийся раскладкой клавиатуры) для системы. (ANSI)

[GetKeyboardLayoutNameW](#)

Извлекает имя идентификатора активного языкового стандарта ввода (ранее называвшийся раскладкой клавиатуры) для системы. (Юникод)

[GetKeyboardState](#)

Копирует состояние 256 виртуальных ключей в указанный буфер.

[GetKeyboardType](#)

Извлекает сведения о текущей клавиатуре.

[GetKeyNameTextA](#)

Извлекает строку, представляющую имя ключа. (ANSI)

[GetKeyNameTextW](#)

Извлекает строку, представляющую имя ключа. (Юникод)

[GetKeyState](#)

Возвращает состояние указанного виртуального ключа. Состояние указывает, находится ли клавиша вверх, вниз или переключается (включено, выключается поочереживание при каждом нажатии клавиши).

[GetLastInputInfo](#)

Извлекает время последнего входного события.

[GetMouseMovePointsEx](#)

Извлекает журнал до 64 предыдущих координат мыши или пера.

[GetRawInputBuffer](#)

Выполняет буферизованное чтение необработанных входных данных.

[GetRawInputData](#)

Извлекает необработанные входные данные с указанного устройства.

[GetRawInputDeviceInfoA](#)

Извлекает сведения о необработанном устройстве ввода. (ANSI)

[GetRawInputDeviceInfoW](#)

Извлекает сведения о необработанном устройстве ввода. (Юникод)

[GetRawInputDeviceList](#)

Перечисляет необработанные устройства ввода, подключенные к системе.

[GetRegisteredRawInputDevices](#)

Извлекает сведения о необработанных устройствах ввода для текущего приложения.

[IsWindowEnabled](#)

Определяет, включено ли указанное окно для ввода с помощью мыши и клавиатуры.

[keybd_event](#)

Синтезирует нажатие клавиши.

[LoadKeyboardLayoutA](#)

Загружает в систему новый идентификатор языкового стандарта ввода (прежнее название — раскладка клавиатуры). (ANSI)

[LoadKeyboardLayoutW](#)

Загружает в систему новый идентификатор языкового стандарта ввода (прежнее название — раскладка клавиатуры). (Юникод)

[MapVirtualKeyA](#)

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. (ANSI)

[MapVirtualKeyExA](#)

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. Функция преобразует коды с помощью языка ввода и идентификатора языкового стандарта. (ANSI)

[MapVirtualKeyExW](#)

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. Функция преобразует коды с помощью языка ввода и идентификатора языкового стандарта. (Юникод)

[MapVirtualKeyW](#)

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. (Юникод)

[mouse_event](#)

Функция `mouse_event` синтезирует движения мыши и нажатия кнопок.

[NEXTRAWINPUTBLOCK](#)

Извлекает расположение следующей структуры в массиве структур `RAWINPUT`.

[OemKeyScan](#)

Сопоставляет коды OEMASCII от 0 до 0x0FF с кодами сканирования OEM и состояниями сдвига. Функция предоставляет сведения, позволяющие программе отправлять текст изготовителя оборудования другой программе путем имитации ввода с клавиатуры.

[RegisterTooltipDismissNotification](#)

Позволяет приложениям или платформам пользовательского интерфейса регистрировать и отменять регистрацию окон для получения уведомлений о закрытии окон всплывающих подсказок.

[RegisterHotKey](#)

Определяет системный горячий ключ.

[RegisterRawInputDevices](#)

Регистрирует устройства, которые предоставляют необработанные входные данные.

[ReleaseCapture](#)

Освобождает захват мыши из окна в текущем потоке и восстанавливает нормальную обработку ввода с помощью мыши.

[SendInput](#)

Синтезирует нажатия клавиш, движения мыши и нажатия кнопок.

[SetActiveWindow](#)

Активирует окно. Окно должно быть присоединено к очереди сообщений вызывающего потока.

[SetCapture](#)

Задает для захвата мыши указанное окно, принадлежащее текущему потоку.

[SetDoubleClickTime](#)

Задает время двойного щелчка мыши.

[SetFocus](#)

Устанавливает фокус клавиатуры в указанное окно. Окно должно быть присоединено к очереди сообщений вызывающего потока.

[SetKeyboardState](#)

Копирует массив состояний клавиш клавиатуры в таблицу состояния ввода-вывода клавиатуры вызывающего потока. Это та же таблица, к которой обращаются функции GetKeyboardState и GetKeyState. Изменения, внесенные в эту таблицу, не влияют на ввод с клавиатуры в любой другой поток.

[SwapMouseButton](#)

Изменяет или восстанавливает значение левой и правой кнопок мыши.

[ToAscii](#)

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы или символы.

[ToAsciiEx](#)

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы или символы. Функция преобразует код с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода.

[ToUnicode](#)

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы Юникода. (ToUnicode)

[ToUnicodeEx](#)

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы Юникода. (ToUnicodeEx)

[TrackMouseEvent](#)

Публикует сообщения, когда указатель мыши покидает окно или наносит указатель мыши на окно в течение указанного периода времени.

[UnloadKeyboardLayout](#)

Выгрузка идентификатора входного языкового стандарта (прежнее название — раскладка клавиатуры).

[UnregisterHotKey](#)

Освобождает горячий ключ, ранее зарегистрированный вызывающим потоком.

[VkKeyScanA](#)

Преобразует символ в соответствующий код виртуальной клавиши и состояние сдвига для текущей клавиатуры. (ANSI)

[VkKeyScanExA](#)

Преобразует символ в соответствующий код виртуального ключа и состояние сдвига. Функция преобразует символ с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода. (ANSI)

[VkKeyScanExW](#)

Преобразует символ в соответствующий код виртуального ключа и состояние сдвига. Функция преобразует символ с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода. (Юникод)

[VkKeyScanW](#)

Преобразует символ в соответствующий код виртуальной клавиши и состояние сдвига для текущей клавиатуры. (Юникод)

Структуры

[HARDWAREINPUT](#)

Содержит сведения о имитированном сообщении, созданном с помощью устройства ввода, отличного от клавиатуры или мыши.

[INPUT](#)

Используется SendInput для хранения информации для синтеза событий ввода, таких как нажатия клавиш, перемещение мыши и щелчки мышью.

[KEYBDINPUT](#)

Содержит сведения о событии имитации клавиатуры.

[LASTINPUTINFO](#)

Содержит время последнего ввода.

[MOUSEINPUT](#)

Содержит сведения о имитированном событии мыши.

[MOUSEMOVEPOINT](#)

Содержит сведения о расположении мыши в координатах экрана.

[RAWHID](#)

Описывает формат необработанных входных данных с устройства HID.

[RAWINPUT](#)

Содержит необработанные входные данные с устройства.

[RAWINPUTDEVICE](#)

Определяет сведения для необработанных устройств ввода.

[RAWINPUTDEVICELIST](#)

Содержит сведения о необработанном устройстве ввода.

[RAWINPUTHEADER](#)

Содержит сведения о заголовке, которые являются частью необработанных входных данных.

[RAWKEYBOARD](#)

Содержит сведения о состоянии клавиатуры.

[RAWMOUSE](#)

Содержит сведения о состоянии мыши.

RID_DEVICE_INFO

Определяет необработанные входные данные, поступающие с любого устройства.

RID_DEVICE_INFO_HID

Определяет необработанные входные данные, поступающие от указанного устройства HID.

RID_DEVICE_INFO_KEYBOARD

Определяет необработанные входные данные, поступающие с указанной клавиатуры.

RID_DEVICE_INFO_MOUSE

Определяет необработанные входные данные, поступающие от указанной мыши.

TRACKMOUSEEVENT

Используется функцией TrackMouseEvent для отслеживания того, когда указатель мыши покидает окно или наводит указатель мыши на окно в течение указанного периода времени.

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Заголовок commctrl.h

Статья 08.06.2023

Этот заголовок используется несколькими технологиями. Дополнительные сведения см. в разделе:

- [Ввод с помощью клавиатуры и мыши](#)
- [Оболочка Windows](#)
- [Элементы управления Windows](#)

commctrl.h содержит следующие программные интерфейсы:

ФУНКЦИИ

[_TrackMouseEvent](#)

Публикует сообщения, когда указатель мыши покидает окно или наносит указатель мыши на окно в течение указанного периода времени. Эта функция вызывает TrackMouseEvent, если она существует, в противном случае эмулирует ее.

[Animate_Close](#)

Закрывает клип AVI. Вы можете использовать этот макрос или отправить сообщение ACM_OPEN явным образом, передав параметры NULL.

[Animate_Create](#)

Создает элемент управления анимацией. Animate_Create вызывает функцию CreateWindow для создания элемента управления анимацией.

[Animate_IsPlaying](#)

Проверяет, воспроизводит ли клип Audio-Video с чередованием (AVI). Вы можете использовать этот макрос или отправить ACM_ISPLAYING сообщение.

[Animate_Open](#)

Открывает клип AVI и отображает первый кадр в элементе управления анимацией. Вы можете использовать этот макрос или отправить сообщение ACM_OPEN явным образом.

[Animate_OpenEx](#)

Открывает клип AVI из ресурса в указанном модуле и отображает его первый кадр в элементе управления анимацией. Вы можете использовать этот макрос или отправить сообщение ACM_OPEN явным образом.

[Animate_Play](#)

Воспроизводит клип AVI в элементе управления анимацией. Элемент управления воспроизводит клип в фоновом режиме, пока поток продолжает выполнение. Вы можете использовать этот макрос или отправить сообщение ACM_PLAY явным образом.

[Animate_Seek](#)

Направляет элемент управления анимацией для отображения определенного кадра клипа AVI. Элемент управления отображает клип в фоновом режиме, пока поток продолжает выполнение. Вы можете использовать этот макрос или отправить сообщение ACM_PLAY явным образом.

[Animate_Stop](#)

Прекращает воспроизведение клипа AVI в элементе управления анимацией. Вы можете использовать этот макрос или отправить сообщение ACM_STOP явным образом.

[Button_GetidealSize](#)

Возвращает размер кнопки, которая лучше всего соответствует тексту и изображению, если имеется список изображений. Вы можете использовать этот макрос или отправить сообщение BCM_GETIDEALSIZE явным образом.

[Button_GetImageList](#)

Возвращает структуру BUTTON_IMAGELIST, описывающую список изображений, заданный для элемента управления "Кнопка". Вы можете использовать этот макрос или отправить сообщение BCM_GETIMAGELIST явным образом.

[Button_GetNote](#)

Возвращает текст заметки, связанной с кнопкой командной ссылки. Вы можете использовать этот макрос или отправить сообщение BCM_GETNOTE явным образом.

[Button_GetNoteLength](#)

Возвращает длину текста заметки, который может отображаться в описании ссылки на команду. Используйте этот макрос или отправьте сообщение BCM_GETNOTELENGTH явным образом.

[Button_GetSplitInfo](#)

Возвращает сведения для указанного элемента управления "Разделенная кнопка". Используйте этот макрос или отправьте сообщение BCM_GETSPLITINFO явным образом.

[Button_GetTextMargin](#)

Возвращает поля, используемые для рисования текста в элементе управления "Кнопка". Вы можете использовать этот макрос или отправить сообщение BCM_GETTEXTMARGIN явным образом.

[Button_SetDropDownState](#)

Задает состояние раскрывающегося списка для указанной кнопки со стилем BS_SPLITBUTTON. Используйте этот макрос или отправьте сообщение BCM_SETDROPDOWNSTATE явным образом.

[Button_SetElevationRequiredState](#)

Задает требуемое состояние повышения прав для указанной кнопки или командной ссылки для отображения значка с повышенными привилегиями. Используйте этот макрос или отправьте сообщение BCM_SETSHIELD явным образом.

[Button_SetImageList](#)

Назначает список изображений элементу управления "Кнопка". Вы можете использовать этот макрос или отправить сообщение BCM_SETIMAGELIST явным образом.

[Button_SetNote](#)

Задает текст заметки, связанной с указанной кнопкой командной ссылки. Вы можете использовать этот макрос или отправить сообщение BCM_SETNOTE явным образом.

[Button_SetSplitInfo](#)

Задает сведения для указанного элемента управления "Разбиение кнопки". Используйте этот макрос или отправьте сообщение BCM_SETSPLITINFO явным образом.

[Button_SetTextMargin](#)

Задает поля для рисования текста в элементе управления "Кнопка". Вы можете использовать этот макрос или отправить сообщение BCM_SETTEXTMARGIN явным образом.

[ComboBox_GetCueBannerText](#)

Возвращает текст баннера подсказки, отображаемый в элементе управления редактированием поля со списком. Используйте этот макрос или отправьте сообщение CB_GETCUEBANNER явным образом.

[ComboBox_GetMinVisible](#)

Возвращает минимальное количество видимых элементов в раскрывающемся списке поля со списком.

[ComboBox_SetCueBannerText](#)

Задает текст баннера подсказки, отображаемый для элемента управления редактированием поля со списком.

[ComboBox_SetMinVisible](#)

Задает минимальное количество видимых элементов в раскрывающемся списке поля со списком.

[CreateMappedBitmap](#)

Создает растровое изображение для использования на панели инструментов.

[CreateStatusWindowA](#)

Создает окно состояния, которое обычно используется для отображения состояния приложения. (ANSI)

[CreateStatusWindowW](#)

Создает окно состояния, которое обычно используется для отображения состояния приложения. (Юникод)

[CreateToolbarEx](#)

Создает окно панели инструментов и добавляет указанные кнопки на панель инструментов.

[CreateUpDownControl](#)

Создает элемент управления вверх-вниз. Примечание. _This функция устарела. Это 16-разрядная функция, и она не может обрабатывать 32-разрядные значения для диапазона и положения.

[DateTime_CloseMonthCal](#)

Закрывает элемент управления "Выбор даты и времени" (DTP). Используйте этот макрос или отправьте сообщение DTM_CLOSEMONTHCAL явным образом.

[DateTime_GetDateTimePickerInfo](#)

Возвращает сведения для указанного элемента управления выбора даты и времени (DTP).

[DateTime_GetIdealSize](#)

Возвращает размер, необходимый для отображения элемента управления без обрезки. Используйте этот макрос или отправьте сообщение DTM_GETIDEALSIZE явным образом.

[DateTime_GetMonthCal](#)

Получает дескриптор элемента управления "Календарь дочернего месяца" средства выбора даты и времени (DTP). Вы можете использовать этот макрос или отправить сообщение DTM_GETMONTHCAL явным образом.

[DateTime_GetMonthCalColor](#)

Возвращает цвет для определенной части календаря месяца в элементе управления "Выбор даты и времени" (DTP). Вы можете использовать этот макрос или отправить сообщение DTM_GETMCCOLOR явным образом.

[DateTime_GetMonthCalFont](#)

Возвращает шрифт, используемый элементом управления "Календарь дочернего месяца" элемента управления "Выбор даты и времени" (DTP). Вы можете использовать этот макрос или отправить сообщение DTM_GETMCFONT явным образом.

[DateTime_GetMonthCalStyle](#)

Возвращает стиль указанного элемента управления выбора даты и времени (DTP). Используйте этот макрос или отправьте сообщение DTM_GETMCSTYLE явным образом.

[DateTime_GetRange](#)

Возвращает текущее минимальное и максимально допустимое системное время для элемента управления выбора даты и времени (DTP). Вы можете использовать этот макрос или отправить сообщение DTM_GETRANGE явным образом.

[DateTime_GetSystemtime](#)

Получает выбранное в данный момент время из элемента управления выбора даты и времени (DTP) и помещает его в указанную структуру SYSTEMTIME. Вы можете использовать этот макрос или отправить сообщение DTM_GETSYSTEMTIME явным образом.

[DateTime_SetFormat](#)

Задает отображение элемента управления выбора даты и времени (DTP) на основе заданной строки формата. Вы можете использовать этот макрос или отправить сообщение DTM_SETFORMAT явным образом.

[DateTime_SetMonthCalColor](#)

Задает цвет для определенной части календаря месяца в элементе управления выбора даты и времени (DTP). Вы можете использовать этот макрос или отправить сообщение DTM_SETMCCOLOR явным образом.

[DateTime_SetMonthCalFont](#)

Задает шрифт, используемый элементом управления "Календарь дочернего месяца" элемента управления "Выбор даты и времени" (DTP). Вы можете использовать этот макрос или явно отправить сообщение DTM_SETMCFONT.

[DateTime_SetMonthCalStyle](#)

Задает стиль для указанного элемента управления выбора даты и времени (DTP). Используйте этот макрос или отправьте сообщение DTM_SETMCSTYLE явным образом.

[DateTime_SetRange](#)

Задает минимальное и максимально допустимое системное время для элемента управления выбора даты и времени (DTP). Вы можете использовать этот макрос или отправить сообщение DTM_SETRANGE явным образом.

[DateTime_SetSystemtime](#)

Задает для элемента управления выбора даты и времени (DTP) заданные дату и время. Вы можете использовать этот макрос или отправить сообщение DTM_SETSYSTEMTIME явным образом.

[DefSubclassProc](#)

Вызывает следующий обработчик в цепочке подклассов окна. Последний обработчик в цепочке подклассов вызывает исходную процедуру окна для окна.

[DrawInsert](#)

Рисует значок вставки в родительском окне указанного списка перетаскивания.

[DrawShadowText](#)

Рисует текст с тенью.

[DrawStatusTextA](#)

Функция DrawStatusText рисует указанный текст в стиле окна состояния с границами. (ANSI)

[DrawStatusTextW](#)

Функция DrawStatusText рисует указанный текст в стиле окна состояния с границами.
(Юникод)

[Edit_EnableSearchWeb](#)

Включает или отключает функцию "Поиск с помощью Bing..." пункт контекстного меню в элементах управления "Редактирование". Вы можете использовать этот макрос или отправить сообщение EM_ENABLESEARCHWEB явным образом.

[Edit_GetCaretIndex](#)

Возвращает символьный индекс расположения курсора для заданного элемента управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_GETCARETINDEX явным образом.

[Edit_GetCueBannerText](#)

Получает текст, отображаемый в виде текстового подсказки или подсказки в элементе управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_GETCUEBANNER явным образом.

[Edit_GetEndOfLine](#)

Возвращает символ конца строки, используемый для содержимого элемента управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_GETENDOFLINE явным образом.

[Edit_GetExtendedStyle](#)

Возвращает расширенные стили, которые в настоящее время используются для данного элемента управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_GETEXTENDEDSTYLE явным образом.

[Edit_GetFileLine](#)

Возвращает текст указанной строки файла (или логической) строки (разделители обтекания текстом игнорируются). Вы можете использовать этот макрос или отправить сообщение EM_GETFILELINE явным образом.

[Edit_GetFileLineCount](#)

Возвращает количество строк файла (или логических) (разделители обтекания текстом игнорируются). Вы можете использовать этот макрос или отправить сообщение EM_GETFILELINECOUNT явным образом.

[Edit_GetFileLineFromChar](#)

Возвращает индекс файловой (или логической) строки текста, которая включает указанный индекс символов (разделители обтекания текста игнорируются). Вы можете использовать этот макрос или отправить сообщение EM_FILELINEFROMCHAR явным образом.

[Edit_GetFileLineIndex](#)

Возвращает индекс файловой (или логической) строки текста на основе указанной видимой строки. Вы можете использовать этот макрос или отправить сообщение EM_FILELINEINDEX явным образом.

[Edit_GetFileLineLength](#)

Возвращает длину файловой (или логической) строки текста из указанного символьного индекса (разделители обтекания текстом игнорируются). Вы можете использовать этот макрос или отправить сообщение EM_FILELINELENGTH явным образом.

[Edit_GetHilite](#)

Этот макрос не реализован. (Edit_GetHilite)

[Edit_GetZoom](#)

Возвращает текущее соотношение масштаба элемента управления редактированием (соотношение масштаба всегда находится в диапазоне от 1/64 до 64). Вы можете использовать этот макрос или отправить сообщение EM_GETZOOM явным образом.

[Edit_HideBalloonTip](#)

Скрывает все всплывающие подсказки, связанные с элементом управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_HIDEBALLOONTIP явным образом.

[Edit_NoSetFocus](#)

Запрещает однострочному элементу управления редактированием получать фокус клавиатуры. Вы можете использовать этот макрос или отправить сообщение EM_NOSETFOCUS явным образом.

[Edit_SearchWeb](#)

Вызывает "Поиск с помощью Bing..." пункт контекстного меню в элементах управления "Редактирование". Вы можете использовать этот макрос или отправить сообщение EM_SEARCHWEB явным образом.

[Edit_SetCaretIndex](#)

Задает индекс символов, по которому нужно найти курсор. Вы можете использовать этот макрос или отправить сообщение EM_SETCARETINDEX явным образом.

[Edit_SetCueBannerText](#)

Задает текст, отображаемый в качестве текстового подсказки или подсказки для элемента управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_SETCUEBANNER явным образом. (Edit_SetCueBannerText)

[Edit_SetCueBannerTextFocused](#)

Задает текст, отображаемый в качестве текстового подсказки или подсказки для элемента управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_SETCUEBANNER явным образом. (Edit_SetCueBannerTextFocused)

[Edit_SetEndOfLine](#)

Задает символ конца строки, используемый для содержимого элемента управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_SETENDOFLINE явным образом.

[Edit_SetExtendedStyle](#)

Задает расширенные стили для редактирования элементов управления с помощью маски стиля. Вы можете использовать этот макрос или отправить сообщение EM_SETEXTENDEDSTYLE явным образом.

[Edit_SetHilite](#)

Этот макрос не реализован. (Edit_SetHilite)

[Edit_SetZoom](#)

Задает текущее соотношение масштаба элемента управления редактированием (соотношение масштаба всегда находится в диапазоне от 1/64 до 64). Вы можете использовать этот макрос или отправить сообщение EM_SETZOOM явным образом.

[Edit_ShowBalloonTip](#)

Отображает всплывающий наконечник, связанный с элементом управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_SHOWBALLOONTIP явным образом.

[Edit_TakeFocus](#)

Принудительное получение фокуса клавиатуры для односторочного элемента управления редактированием. Вы можете использовать этот макрос или отправить сообщение EM_TAKEFOCUS явным образом.

[FIRST_IPADDRESS](#)

Извлекает значение поля 0 из упакованного IP-адреса, полученного с сообщением IPM_GETADDRESS.

[FlatSB_EnableScrollBar](#)

Включает или отключает одну или обе кнопки направления плоской полосы прокрутки. Если для окна не инициализированы плоские полосы прокрутки, эта функция вызывает стандартную функцию EnableScrollBar.

[FlatSB_GetScrollInfo](#)

Возвращает сведения для плоской полосы прокрутки. Если для окна не инициализированы плоские полосы прокрутки, эта функция вызывает стандартную функцию GetScrollInfo.

[FlatSB_GetScrollPos](#)

Получает положение большого пальца на плоской полосе прокрутки. Если для окна не инициализированы плоские полосы прокрутки, эта функция вызывает стандартную функцию GetScrollPos.

[FlatSB_GetScrollProp](#)

Возвращает свойства для плоской полосы прокрутки. Эта функция также может использоваться для определения того, был ли вызван InitializeFlatSB для этого окна.

[FlatSB_GetScrollPropPtr](#)

Возвращает свойства для плоской полосы прокрутки.

[FlatSB_GetScrollRange](#)

Возвращает диапазон прокрутки для плоской полосы прокрутки. Если для окна не инициализированы плоские полосы прокрутки, эта функция вызывает стандартную функцию GetScrollRange.

[FlatSB_SetScrollInfo](#)

Задает сведения для плоской полосы прокрутки. Если для окна не инициализированы плоские полосы прокрутки, эта функция вызывает стандартную функцию SetScrollInfo.

[FlatSB_SetScrollPos](#)

Задает текущее положение большого пальца на плоской полосе прокрутки. Если для окна не инициализированы плоские полосы прокрутки, эта функция вызывает стандартную функцию SetScrollPos.

[FlatSB_SetScrollProp](#)

Задает свойства для плоской полосы прокрутки.

[FlatSB_SetScrollRange](#)

Задает диапазон прокрутки плоской полосы прокрутки. Если для окна не инициализированы плоские полосы прокрутки, эта функция вызывает стандартную функцию SetScrollRange.

[FlatSB_ShowScrollBar](#)

Отображает или скрывает плоскую полосу прокрутки. Если для окна не инициализированы плоские полосы прокрутки, эта функция вызывает стандартную функцию ShowScrollBar.

[FORWARD_WM_NOTIFY](#)

Отправляет или публикует сообщение WM_NOTIFY.

[FOURTH_IPADDRESS](#)

Извлекает значение поля 3 из упакованного IP-адреса, полученного с сообщением IPM_GETADDRESS.

[GetEffectiveClientRect](#)

Вычисляет размеры прямоугольника в клиентской области, содержащей все указанные элементы управления.

[GetMUILanguage](#)

Возвращает язык, используемый в настоящее время общими элементами управления для определенного процесса.

[GetWindowSubclass](#)

Извлекает эталонные данные для обратного вызова указанного подкласса окна.

[HANDLE_WM_NOTIFY](#)

Вызывает функцию, которая обрабатывает сообщение WM_NOTIFY.

[Header_ClearAllFilters](#)

Очищает все фильтры для заданного элемента управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_CLEARFILTER явным образом.

[Header_ClearFilter](#)

Очищает фильтр для заданного элемента управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_CLEARFILTER явным образом.

[Header_CreateDragImage](#)

Создает прозрачную версию изображения элемента в существующем элементе управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_CREATEDRAGIMAGE явным образом.

[Header_DeleteItem](#)

Удаляет элемент из элемента управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_DELETEITEM явным образом.

[Header_EditFilter](#)

Перемещает фокус ввода в поле редактирования при нажатии кнопки фильтра.

[Header_GetBitmapMargin](#)

Возвращает ширину поля (в пикселях) растрового изображения в существующем элементе управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_GETBITMAPMARGIN явным образом.

[Header_GetFocusedItem](#)

Возвращает элемент в элементе управления заголовком, который имеет фокус. Используйте этот макрос или отправьте сообщение HDM_GETFOCUSITEM явным образом.

[Header_GetImageList](#)

Возвращает дескриптор списка изображений, который был задан для существующего элемента управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_GETIMAGELIST явным образом.

[Header_GetItem](#)

Возвращает сведения об элементе в элементе управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_GETITEM явным образом.

[Header_GetItemCount](#)

Возвращает количество элементов в элементе управления "Заголовок". Вы можете использовать этот макрос или отправить сообщение HDM_GETITEMCOUNT явным образом.

[Header_GetItemDropDownRect](#)

Возвращает координаты кнопки раскрывающегося списка для указанного элемента в элементе управления "Заголовок". Элемент управления "Заголовок" должен иметь тип HDF_SPLITBUTTON. Используйте этот макрос или отправьте сообщение HDM_GETITEMDROPDOWNRECT явным образом.

[Header_GetItemRect](#)

Возвращает ограничивающий прямоугольник для заданного элемента в элементе управления "Заголовок". Вы можете использовать этот макрос или отправить сообщение HDM_GETITEMRECT явным образом.

[Header_GetOrderArray](#)

Возвращает текущий порядок элементов слева направо в элементе управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_GETORDERARRAY явным образом.

[Header_GetOverflowRect](#)

Возвращает координаты области раскрывающегося списка для указанного элемента управления заголовком. Элемент управления "Заголовок" должен иметь тип HDF_SPLITBUTTON. Используйте этот макрос или отправьте сообщение HDM_GETOVERFLOWRECT явным образом.

[Header_GetStatelImageList](#)

Возвращает дескриптор списка изображений, который был задан для существующего состояния элемента управления заголовком.

[Header_GetUnicodeFormat](#)

Возвращает флаг символьного формата Юникода для элемента управления . Вы можете использовать этот макрос или отправить сообщение HDM_GETUNICODEFORMAT явным образом.

[Header_InsertItem](#)

Вставляет новый элемент в элемент управления "Заголовок". Вы можете использовать этот макрос или отправить сообщение HDM_INSERTITEM явным образом.

[Header_Layout](#)

Возвращает правильный размер и положение элемента управления заголовком в родительском окне. Вы можете использовать этот макрос или отправить сообщение HDM_LAYOUT явным образом.

[Header_OrderToIndex](#)

Извлекает значение индекса для элемента на основе его порядка в элементе управления "Заголовок". Вы можете использовать этот макрос или отправить сообщение HDM_ORDERTOINDEX явным образом.

[Header_SetBitmapMargin](#)

Задает ширину поля для растрового изображения в существующем элементе управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_SETBITMAPMARGIN явным образом.

[Header_SetFilterChangeTimeout](#)

Задает интервал времени ожидания между временем изменения в атрибутах фильтра и публикацией уведомления HDN_FILTERCHANGE. Вы можете использовать этот макрос или отправить сообщение HDM_SETFILTERCHANGETIMEOUT явным образом.

[Header_SetFocusedItem](#)

Устанавливает фокус на указанный элемент в элементе управления "Заголовок". Используйте этот макрос или отправьте сообщение HDM_SETFOCUSEDITEM явным образом.

[Header_SetHotDivider](#)

Изменяет цвет разделителя между элементами заголовка, чтобы указать назначение внешней операции перетаскивания. Вы можете использовать этот макрос или отправить сообщение HDM_SETHOTDIVIDER явным образом.

[Header_SetImageList](#)

Назначает список изображений существующему элементу управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_SETIMAGELIST явным образом.

[Header_SetItem](#)

Задает атрибуты указанного элемента в элементе управления заголовком. Вы можете использовать этот макрос или отправить сообщение HDM_SETITEM явным образом.

[Header_SetOrderArray](#)

Задает порядок элементов заголовка слева направо. Вы можете использовать этот макрос или отправить сообщение HDM_SETORDERARRAY явным образом.

[Header_SetStateImageList](#)

Назначает список изображений существующему состоянию элемента управления заголовком.

[Header_SetUnicodeFormat](#)

Задает флаг символьного формата ЮНИКОДа для элемента управления .

[HIMAGELIST_QueryInterface](#)

Извлекает указатель на объект [IImageList](#) или [IImageList2](#), соответствующий дескриптору HIMAGELIST списка изображений.

[ImageList_Add](#)

Добавляет изображение или изображения в список изображений. ([ImageList_Add](#))

[ImageList_AddIcon](#)

Добавляет значок или курсор в список изображений. [ImageList_AddIcon](#) вызывает функцию [ImageList_ReplacerIcon](#).

[ImageList_AddMasked](#)

Добавляет изображение или изображения в список изображений, создавая маску из указанного растрового изображения. ([ImageList_AddMasked](#))

[ImageList_BeginDrag](#)

Начинает перетаскивание изображения. ([ImageList_BeginDrag](#))

[ImageList_Copy](#)

Копирует изображения в заданном списке изображений.

[ImageList_Create](#)

Создает новый список изображений.

[ImageList_Destroy](#)

Уничтожает список изображений.

[ImageList_DragEnter](#)

Отображает изображение перетаскивания в указанной позиции в окне.

[ImageList_DragLeave](#)

Разблокирует указанное окно и скрывает изображение перетаскивания, позволяя обновить окно.

[ImageList_DragMove](#)

Перемещает перетаскиваемое изображение во время операции перетаскивания. Эта функция обычно вызывается в ответ на WM_MOUSEMOVE сообщение. ([ImageList_DragMove](#))

[ImageList_DragShowNolock](#)

Показывает или скрывает перетаскиваемое изображение. ([ImageList_DragShowNolock](#))

[ImageList_Draw](#)

Рисует элемент списка изображений в указанном контексте устройства. ([ImageList_Draw](#))

[ImageList_DrawEx](#)

Рисует элемент списка изображений в указанном контексте устройства. Функция использует указанный стиль рисования и смешивает изображение с указанным цветом.

[ImageList_DrawIndirect](#)

Рисует изображение списка изображений на основе структуры IMAGELISTDRAWPARAMS.

[ImageList_Duplicate](#)

Создает дубликат существующего списка изображений.

[ImageList_EndDrag](#)

Завершает операцию перетаскивания. ([ImageList_EndDrag](#))

[ImageList_ExtractIcon](#)

Вызывает функцию [ImageList_GetIcon](#) для создания значка или курсора на основе изображения и маски в списке изображений.

[ImageList_GetBkColor](#)

Извлекает текущий цвет фона для списка изображений.

[ImageList_GetDragImage](#)

Извлекает список временных изображений, используемый для перетаскивания изображения. Функция также извлекает текущую позицию перетаскивания и смещение изображения перетаскивания относительно позиции перетаскивания.

[ImageList_GetIcon](#)

Создает значок из изображения и маски в списке изображений.

[ImageList_GetIconSize](#)

Извлекает размеры изображений в списке изображений. Все изображения в списке изображений имеют одинаковые размеры.

[ImageList_GetImageCount](#)

Извлекает количество изображений в списке изображений.

[ImageList_GetImageInfo](#)

Извлекает сведения об изображении.

[ImageList_LoadBitmap](#)

Вызывает функцию `ImageList_LoadImage` для создания списка изображений из указанного ресурса растрового изображения.

[ImageList_LoadImageA](#)

Создает список изображений на основе указанного растрового изображения. (ANSI)

[ImageList_LoadImageW](#)

Создает список изображений на основе указанного растрового изображения. (Юникод)

[ImageList_Merge](#)

Создает новый образ путем объединения двух существующих образов. Функция также создает новый список изображений, в котором будет храниться изображение.

[ImageList_Read](#)

Считывает список изображений из потока.

[ImageList_ReadEx](#)

Считывает список изображений из потока и возвращает интерфейс `IImageList` в список изображений.

[ImageList_Remove](#)

Удаляет изображение из списка изображений. (`ImageList_Remove`)

[ImageList_RemoveAll](#)

Вызывает функцию `ImageList_Remove` для удаления всех изображений из списка изображений.

[ImageList_Replace](#)

Заменяет изображение в списке образов новым изображением. (`ImageList_Replace`)

[ImageList_ReplacerIcon](#)

Заменяет изображение значком или курсором. (`ImageList_ReplacerIcon`)

[ImageList_SetBkColor](#)

Задает цвет фона для списка изображений. Эта функция работает только при добавлении значка или использовании `ImageList_AddMasked` с черно-белым растровым изображением. Без маски рисуется все изображение; поэтому цвет фона не виден.

[ImageList_SetDragCursorImage](#)

Создает новое изображение перетаскивания путем объединения указанного изображения (обычно это изображение курсора мыши) с текущим изображением перетаскивания.

[ImageList_SetIconSize](#)

Задает размеры изображений в списке изображений и удаляет все изображения из списка. (`ImageList_SetIconSize`)

[ImageList_SetImageCount](#)

Изменяет размер существующего списка изображений. (`ImageList_SetImageCount`)

[ImageList_SetOverlayImage](#)

Добавляет указанное изображение в список изображений, используемых в качестве масок наложения. Список изображений может содержать до четырех масок наложения в версии 4.70 и более ранних и до 15 в версии 4.71. Функция назначает индекс маски наложения указанному изображению.

[ImageList_Write](#)

Записывает список изображений в поток. (`ImageList_Write`)

[ImageList_WriteEx](#)

Записывает список изображений в поток. ([ImageList_WriteEx](#))

[INDEXTOOVERLAYMASK](#)

Подготавливает индекс маски наложения, чтобы функция [ImageList_Draw](#) пользовалась ею.

[INDEXTOSTATEIMAGEMASK](#)

Подготавливает индекс изображения состояния, чтобы элемент управления представлением в виде дерева или элементом управления представлением списка можно было использовать индекс для получения изображения состояния для элемента.

[InitCommonControls](#)

Регистрирует и инициализирует некоторые общие классы окна управления. Эта функция является устаревшей. Новые приложения должны использовать функцию [InitCommonControlsEx](#).

[InitCommonControlsEx](#)

Обеспечивает загрузку библиотеки DLL общего элемента управления (Comctl32.dll) и регистрирует определенные общие классы элементов управления из библиотеки DLL. Приложение должно вызвать эту функцию перед созданием общего элемента управления.

[InitializeflatSB](#)

Инициализирует плоские полосы прокрутки для определенного окна.

[InitMUILanguage](#)

Позволяет приложению указать язык, который будет использоваться с общими элементами управления, отличными от системного языка.

[LBItemFromPt](#)

Извлекает индекс элемента в указанной точке списка.

[ListView_ApproximateViewRect](#)

Вычисляет приблизительную ширину и высоту, необходимые для отображения заданного количества элементов. Вы можете использовать этот макрос или отправить сообщение LVM_APPROXIMATEVIEWRECT явным образом.

[ListView_Arrange](#)

Упорядочивает элементы в представлении значков. Вы можете использовать этот макрос или отправить сообщение LVM_ARRANGE явным образом.

[ListView_CancelEditLabel](#)

Отменяет операцию редактирования текста элемента. Вы можете использовать этот макрос или отправить сообщение LVM_CANCELEDITLABEL явным образом.

[ListView_CreateDragImage](#)

Создает список изображений перетаскивания для указанного элемента. Вы можете использовать этот макрос или отправить сообщение LVM_CREATEDRAGIMAGE явным образом.

[ListView_DeleteAllItems](#)

Удаляет все элементы из элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_DELETEALLITEMS явным образом.

[ListView_DeleteColumn](#)

Удаляет столбец из элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_DELETECOLUMN явным образом.

[ListView_DeleteItem](#)

Удаляет элемент из элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_DELETEITEM явным образом.

[ListView_EditLabel](#)

Начинается редактирование текста указанного элемента представления списка на месте. Сообщение неявно выбирает и фокусирует указанный элемент. Вы можете использовать этот макрос или отправить сообщение LVM_EDITLABEL явным образом.

[ListView_EnableGroupView](#)

Включает или отключает отображение элементов в элементе управления представления списка в виде группы. Вы можете использовать этот макрос или отправить сообщение LVM_ENABLEGROUPVIEW явным образом.

[ListView_EnsureVisible](#)

Гарантирует, что элемент представления списка полностью или частично отображается, при необходимости прокручивая элемент управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_ENSUREVISIBLE явным образом.

[ListView_FindItem](#)

Выполняет поиск элемента представления списка с указанными характеристиками. Вы можете использовать этот макрос или отправить сообщение LVM_FINDITEM явным образом.

[ListView_GetBkColor](#)

Возвращает цвет фона элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETBKCOLOR явным образом.

[ListView_GetBkImage](#)

Возвращает фоновое изображение в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETBKIMAGE явным образом.

[ListView_GetCallbackMask](#)

Возвращает маску обратного вызова для элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETCALLBACKMASK явным образом.

[ListView_GetCheckState](#)

Определяет, выбран ли элемент в элементе управления представлением списка. Его следует использовать только для элементов управления представлением списка со стилем LVS_EX_CHECKBOXES.

[ListView_GetColumn](#)

Возвращает атрибуты столбца элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETCOLUMN явным образом.

[ListView_GetColumnOrderArray](#)

Возвращает текущий порядок столбцов слева направо в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETCOLUMNORDERARRAY явным образом.

[ListView_GetColumnWidth](#)

Возвращает ширину столбца в представлении отчета или списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETCOLUMNWIDTH явным образом.

[ListView_GetCountPerPage](#)

Вычисляет количество элементов, которые могут поместиться вертикально в видимой области элемента управления представлением списка в представлении списка или отчета. Учитываются только полностью видимые элементы. Вы можете использовать этот макрос или отправить сообщение LVM_GETCOUNTPERPAGE явным образом.

[ListView_GetEditControl](#)

Получает дескриптор элемента управления редактированием, используемый для редактирования текста элемента представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETEDITCONTROL явным образом.

[ListView_GetEmptyText](#)

Получает текст, предназначенный для отображения, когда элемент управления представлением списка отображается пустым. Используйте этот макрос или отправьте сообщение LVM_GETEMPTYTEXT явным образом.

[ListView_GetExtendedListViewStyle](#)

Возвращает расширенные стили, которые в настоящее время используются для данного элемента управления представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETEXTENDEDLISTVIEWSTYLE явным образом.

[ListView_GetFocusedGroup](#)

Возвращает группу с фокусом. Используйте этот макрос или отправьте сообщение LVM_GETFOCUSEDGROUP явным образом.

[ListView_GetFooterInfo](#)

Получает сведения в нижнем колонтикле указанного элемента управления представлением списка. Используйте этот макрос или отправьте сообщение LVM_GETFOOTERINFO явным образом.

[ListView_GetFooterItem](#)

Получает сведения о нижнем колонтикле для указанного элемента управления представлением списка. Используйте этот макрос или отправьте сообщение LVM_GETFOOTERITEM явным образом.

[ListView_GetFooterItemRect](#)

Возвращает координаты нижнего колонтикла для указанного элемента в элементе управления представлением списка. Используйте этот макрос или отправьте сообщение LVM_GETFOOTERITEMRECT явным образом.

[ListView_GetFooterRect](#)

Возвращает координаты нижнего колонтикла для указанного элемента управления представлением списка. Используйте этот макрос или отправьте сообщение LVM_GETFOOTERRECT явным образом.

[ListView_GetGroupCount](#)

Возвращает количество групп. Вы можете использовать этот макрос или отправить сообщение LVM_GETGROUPCOUNT явным образом.

[ListView_GetGroupHeaderImageList](#)

Возвращает список изображений заголовка группы, заданный для существующего элемента управления представлением списка.

[ListView_GetGroupInfo](#)

Возвращает сведения о группе. Вы можете использовать этот макрос или отправить сообщение LVM_GETGROUPINFO явным образом.

[ListView_GetGroupInfoByIndex](#)

Получает сведения об указанной группе. Используйте этот макрос или отправьте сообщение LVM_GETGROUPINFOBYINDEX явным образом.

[ListView_GetGroupMetrics](#)

Возвращает сведения о отображении групп. Вы можете использовать этот макрос или отправить сообщение LVM_GETGROUPEMETRICS явным образом.

[ListView_GetGroupRect](#)

Возвращает прямоугольник для указанной группы. Используйте этот макрос или отправьте сообщение LVM_GETGROUPRECT явным образом.

[ListView_GetGroupState](#)

Возвращает состояние для указанной группы. Используйте этот макрос или отправьте сообщение LVM_GETGROUPSTATE явным образом.

[ListView_GetHeader](#)

Возвращает дескриптор для элемента управления заголовком, используемого элементом управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETHEADER явным образом.

[ListView_GetHotCursor](#)

Возвращает объект HCURSOR, используемый при наведении указателя на элемент при включенном горячем отслеживании. Вы можете использовать этот макрос или отправить сообщение LVM_GETHOTCURSOR явным образом.

[ListView_GetHotItem](#)

Возвращает индекс горячего элемента. Вы можете использовать этот макрос или отправить сообщение LVM_GETHOTITEM явным образом.

[ListView_GetHoverTime](#)

Возвращает время, в течение которого курсор мыши должен наведите указатель мыши на элемент, прежде чем он будет выбран. Вы можете использовать этот макрос или отправить сообщение LVM_GETHOVERTIME явным образом.

[ListView_GetImageList](#)

Получает дескриптор списка изображений, используемый для рисования элементов представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETIMAGELIST явным образом.

[ListView_GetInsertMark](#)

Возвращает позицию точки вставки. Вы можете использовать этот макрос или отправить сообщение LVM_GETINSERTMARK явным образом.

[ListView_GetInsertMarkColor](#)

Возвращает цвет точки вставки. Вы можете использовать этот макрос или отправить сообщение LVM_GETINSERTMARKCOLOR явным образом.

[ListView_GetInsertMarkRect](#)

Возвращает прямоугольник, ограничивающий точку вставки. Вы можете использовать этот макрос или отправить сообщение LVM_GETINSERTMARKRECT явным образом.

[ListView_GetSearchString](#)

Возвращает строку добавочного поиска элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETSEARCHSTRING явным образом.

[ListView_GetItem](#)

Возвращает некоторые или все атрибуты элемента представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETITEM явным образом.

[ListView_GetItemCount](#)

Возвращает количество элементов в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETITEMCOUNT явным образом.

[ListView_GetItemIndexRect](#)

Возвращает ограничивающий прямоугольник для всего или части подэлемента в текущем представлении указанного элемента управления list-view. Используйте этот макрос или отправьте сообщение LVM_GETITEMINDEXRECT явным образом.

[ListView_GetItemPosition](#)

Возвращает положение элемента представления списка. Вы можете использовать этот макрос или явно отправить LVM_GETITEMPOSITION сообщение.

[ListView_GetItemRect](#)

Возвращает ограничивающий прямоугольник для всего элемента в текущем представлении или его части. Вы можете использовать этот макрос или отправить сообщение LVM_GETITEMRECT явным образом.

[ListView_GetItemSpacing](#)

Определяет интервал между элементами в элементе управления представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETITEMSPACING явным образом.

[ListView_GetItemState](#)

Возвращает состояние элемента представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETITEMSTATE явным образом.

[ListView_GetItemText](#)

Возвращает текст элемента или подэлемента представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETITEMTEXT явным образом.

[ListView_GetNextItem](#)

Выполняет поиск элемента представления списка, который имеет указанные свойства и имеет указанную связь с указанным элементом. Вы можете использовать этот макрос или отправить сообщение LVM_GETNEXTITEM явным образом.

[ListView_GetNextItemIndex](#)

Возвращает индекс элемента в определенном элементе управления представления списка, который имеет указанные свойства и связь с другим конкретным элементом. Используйте этот макрос или отправьте сообщение LVM_GETNEXTITEMINDEX явным образом.

[ListView_GetNumberOfWorkAreas](#)

Возвращает количество рабочих областей в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETNUMBEROFWORKAREAS явным образом.

[ListView_GetOrigin](#)

Возвращает источник текущего представления для элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETORIGIN явным образом.

[ListView_GetOutlineColor](#)

Возвращает цвет границы элемента управления представлением списка, если задан стиль LVS_EX_BORDERSELECT расширенного окна. Вы можете использовать этот макрос или отправить сообщение LVM_GETOUTLINECOLOR явным образом.

[ListView_GetSelectedColumn](#)

Возвращает целое число, указывающее выбранный столбец. Вы можете использовать этот макрос или отправить сообщение LVM_GETSELECTEDCOLUMN явным образом.

[ListView_GetSelectedCount](#)

Определяет количество выбранных элементов в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETSELECTEDCOUNT явным образом.

[ListView_GetSelectionMark](#)

Возвращает знак выбора из элемента управления "Представление списка". Вы можете использовать этот макрос или явно отправить LVM_GETSELECTIONMARK сообщение.

[ListView_GetStringWidth](#)

Определяет ширину указанной строки с помощью текущего шрифта указанного элемента управления list-view. Вы можете использовать этот макрос или отправить сообщение LVM_GETSTRINGWIDTH явным образом.

[ListView_GetSubItemRect](#)

Возвращает сведения о прямоугольнике, который окружает подэлемент в элементе управления представлением списка.

[ListView_GetTextBkColor](#)

Возвращает цвет фона текста элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETTEXTBKCOLOR явным образом.

[ListView_GetTextColor](#)

Возвращает цвет текста элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETTEXTCOLOR явным образом.

[ListView_GetTileInfo](#)

Возвращает сведения о плитке в элементе управления "Представление списка". Вы можете использовать этот макрос или отправить сообщение LVM_GETTILEINFO явным образом.

[ListView_GetTileViewInfo](#)

Возвращает сведения об элементе управления в представлении списка в представлении плитки. Вы можете использовать этот макрос или отправить сообщение LVM_GETTILEVIEWINFO явным образом.

[ListView_GetToolTips](#)

Возвращает элемент управления подсказкой, который используется элементом управления представлением списка для отображения подсказок. Вы можете использовать этот макрос или отправить сообщение LVM_GETTOOLTIPS явным образом.

[ListView_GetTopIndex](#)

Возвращает индекс самого верхнего видимого элемента в представлении списка или отчета. Вы можете использовать этот макрос или отправить сообщение LVM_GETTOPINDEX явным образом.

[ListView_GetUnicodeFormat](#)

Возвращает флаг символьного формата Юникода для элемента управления . Вы можете использовать этот макрос или отправить сообщение LVM_GETUNICODEFORMAT явным образом.

[ListView_GetView](#)

Возвращает текущее представление элемента управления list-view. Вы можете использовать этот макрос или отправить сообщение LVM_GETVIEW явным образом.

[ListView_GetViewRect](#)

Возвращает ограничивающий прямоугольник всех элементов в элементе управления "Представление списка". Представление списка должно находиться в режиме значка или небольшого значка. Вы можете использовать этот макрос или отправить сообщение LVM_GETVIEWRECT явным образом.

[ListView_GetWorkAreas](#)

Возвращает рабочие области из элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_GETWORKAREAS явным образом.

[ListView_HasGroup](#)

Определяет, имеет ли элемент управления представление списка указанную группу. Вы можете использовать этот макрос или отправить сообщение LVM_HASGROUP явным образом.

[ListView_HitTest](#)

Определяет, какой элемент представления списка (если таковой имеется) находится в указанной позиции. Вы можете использовать этот макрос или отправить сообщение LVM_HITTEST явным образом. (ListView_HitTest)

[ListView_HitTestEx](#)

Определяет, какой элемент представления списка (если таковой имеется) находится в указанной позиции. Вы можете использовать этот макрос или отправить сообщение LVM_HITTEST явным образом. (ListView_HitTestEx)

[ListView_InsertColumn](#)

Вставляет новый столбец в элемент управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_INSERTCOLUMN явным образом.

[ListView_InsertGroup](#)

Вставляет группу в элемент управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_INSERTGROUP явным образом.

[ListView_InsertGroupSorted](#)

Вставляет группу в упорядоченный список групп. Вы можете использовать этот макрос или отправить сообщение LVM_INSERTGROUPSORTED явным образом.

[ListView_InsertItem](#)

Вставляет новый элемент в элемент управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_INSERTITEM явным образом.

[ListView_InsertMarkHitTest](#)

Извлекает точку вставки, ближайшую к указанной точке. Вы можете использовать этот макрос или отправить сообщение LVM_INSERTMARKHITTEST явным образом.

[ListView_IsGroupViewEnabled](#)

Проверяет, включен ли в элементе управления представление списка представление группы. Вы можете использовать этот макрос или отправить сообщение LVM_ISGROUPVIEWENABLED явным образом.

[ListView_IsItemVisible](#)

Указывает, отображается ли элемент в элементе управления представлением списка. Используйте этот макрос или отправьте сообщение LVM_ISITEMVISIBLE явным образом.

[ListView_MapIDToIndex](#)

Сопоставляет идентификатор элемента с индексом. Вы можете использовать этот макрос или отправить сообщение LVM_MAPIDTOINDEX явным образом.

[ListView_MapIndexToID](#)

Сопоставляет индекс элемента с уникальным идентификатором. Вы можете использовать этот макрос или отправить сообщение LVM_MAPINDEXTOID явным образом.

[ListView_MoveGroup](#)

Этот макрос не реализован. (ListView_MoveGroup)

[ListView_MoveltemToGroup](#)

Этот макрос не реализован. (ListView_MoveltemToGroup)

[ListView_RedrawItems](#)

Заставляет элемент управления представлением списка перерисовывать диапазон элементов. Вы можете использовать этот макрос или отправить сообщение LVM_REDRAWITEMS явным образом.

[ListView_RemoveAllGroups](#)

Удаляет все группы из элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_REMOVEALLGROUPS явным образом.

[ListView_RemoveGroup](#)

Удаляет группу из элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM REMOVEGROUP явным образом.

[ListView_Scroll](#)

Прокручивает содержимое элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SCROLL явным образом.

[ListView_SetBkColor](#)

Задает цвет фона элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETBKCOLOR явным образом.

[ListView_SetBkImage](#)

Задает фоновое изображение в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETBKIMAGE явным образом.

[ListView_SetCallbackMask](#)

Изменяет маску обратного вызова для элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETCALLBACKMASK явным образом.

[ListView_SetCheckState](#)

Выбирает или отменяет выбор элемента в элементе управления представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETITEMSTATE явным образом.

[ListView_SetColumn](#)

Задает атрибуты столбца представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETCOLUMN явным образом.

[ListView_SetColumnOrderArray](#)

Задает порядок столбцов в представлении списка слева направо. Вы можете использовать этот макрос или отправить сообщение LVM_SETCOLUMNORDERARRAY явным образом.

[ListView_SetColumnWidth](#)

Используется для изменения ширины столбца в представлении отчета или ширины всех столбцов в режиме представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETCOLUMNWIDTH явным образом.

[ListView_SetExtendedListViewStyle](#)

Задает расширенные стили для элементов управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETEXTENDEDLISTVIEWSTYLE явным образом.

[ListView_SetExtendedListViewStyleEx](#)

Задает расширенные стили для элементов управления представлением списка с помощью маски стиля. Вы можете использовать этот макрос или отправить сообщение LVM_SETEXTENDEDLISTVIEWSTYLE явным образом.

[ListView_SetGroupHeaderImageList](#)

Назначает список изображений заголовку группы элемента управления представлением списка.

[ListView_SetGroupInfo](#)

Задает сведения о группе. Вы можете использовать этот макрос или отправить сообщение LVM_SetGroupInfo явным образом.

[ListView_SetGroupMetrics](#)

Задает сведения о отображении групп. Вы можете использовать этот макрос или отправить сообщение LVM_SetGroupMetrics явным образом.

[ListView_SetGroupState](#)

Задает состояние для указанной группы.

[ListView_SetHotCursor](#)

Задает HCURSOR, который используется элементом управления представлением списка при наведении указателя на элемент при включенном горячем отслеживании. Вы можете использовать этот макрос или отправить сообщение LVM_SetHotCursor явным образом. Чтобы проверка, включено ли горячее отслеживание, вызовите SystemParametersInfo.

[ListView_SetHotItem](#)

Задает горячий элемент в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SetHotItem явным образом.

[ListView_SetHoverTime](#)

Задает время, в течение которого курсор мыши должен наведен на элемент, прежде чем он будет выбран. Вы можете использовать этот макрос или отправить сообщение LVM_SETHOVERTIME явным образом.

[ListView_SetIconSpacing](#)

Задает интервал между значками в элементах управления представлением списка, для которых задан стиль LVS_ICON. Вы можете использовать этот макрос или отправить сообщение LVM_SETICONSPACING явным образом.

[ListView_SetImageList](#)

Назначает список изображений элементу управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETIMAGELIST явным образом.

[ListView_SetInfoTip](#)

Задает текст подсказки. Вы можете использовать этот макрос или отправить сообщение LVM_SETINFOTIP явным образом.

[ListView_SetInsertMark](#)

Задает точку вставки в определенную позицию. Вы можете использовать этот макрос или отправить сообщение LVM_SETINSERTMARK явным образом.

[ListView_SetInsertMarkColor](#)

Задает цвет точки вставки. Вы можете использовать этот макрос или отправить сообщение LVM_SETINSERTMARKCOLOR явным образом.

[ListView_SetItem](#)

Задает некоторые или все атрибуты элемента представления списка. Вы также можете использовать ListView_SetItem для задания текста подэлемента. Вы можете использовать этот макрос или отправить сообщение LVM_SETITEM явным образом.

[ListView_SetItemCount](#)

Заставляет элемент управления представлением списка выделять память для указанного количества элементов. Вы можете использовать этот макрос или отправить сообщение LVM_SETITEMCOUNT явным образом.

[ListView_SetItemCountEx](#)

Задает виртуальное число элементов в представлении виртуального списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETITEMCOUNT явным образом.

[ListView_SetItemIndexState](#)

Задает состояние указанного элемента представления списка. Используйте этот макрос или отправьте сообщение LVM_SETITEMINDEXSTATE явным образом.

[ListView_SetItemPosition](#)

Перемещает элемент в указанное положение в элементе управления представлением списка (в представлении значков или небольших значков). Вы можете использовать этот макрос или отправить сообщение LVM_SETITEMPOSITION явным образом.

[ListView_SetItemPosition32](#)

Перемещает элемент в указанное положение в элементе управления представлением списка (в представлении значков или небольших значков).

[ListView_SetItemState](#)

Изменяет состояние элемента в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETITEMSTATE явным образом.

[ListView_SetItemText](#)

Изменяет текст элемента представления списка или подэлемента. Вы можете использовать этот макрос или отправить сообщение LVM_SETITEMTEXT явным образом.

[ListView_SetOutlineColor](#)

Задает цвет границы элемента управления представлением списка, если задан LVS_EX_BORDERSELECT расширенный стиль окна. Вы можете использовать этот макрос или отправить сообщение LVM_SETOUTLINECOLOR явным образом.

[ListView_SetSelectedColumn](#)

Задает индекс выбранного столбца. Вы можете использовать этот макрос или отправить сообщение LVM_SETSELECTEDCOLUMN явным образом.

[ListView_SetSelectionMark](#)

Задает метку выбора в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETSELECTIONMARK явным образом.

[ListView_SetTextBkColor](#)

Задает цвет фона текста в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETTEXTBKCOLOR явным образом.

[ListView_SetTextColor](#)

Задает цвет текста элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETTEXTCOLOR явным образом.

[ListView_SetTileInfo](#)

Задает сведения для существующей плитки элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETTILEINFO явным образом.

[ListView_SetTileViewInfo](#)

Задает сведения, которые элемент управления представлением списка использует в представлении плитки. Вы можете использовать этот макрос или отправить сообщение LVM_SETTILEVIEWINFO явным образом.

[ListView_SetToolTips](#)

Задает элемент управления подсказкой, который будет использоваться элементом управления представлением списка для отображения подсказок. Вы можете использовать этот макрос или отправить сообщение LVM_SETOOLTIPS явным образом.

[ListView_SetUnicodeFormat](#)

Задает флаг символьного формата Юникода для элемента управления .
(ListView_SetUnicodeFormat)

[ListView_SetView](#)

Задает представление элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETVIEW явным образом.

[ListView_SetWorkAreas](#)

Задает рабочие области в элементе управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_SETWORKAREAS явным образом.

[ListView_SortGroups](#)

Использует функцию сравнения, определяемую приложением, для сортировки групп по идентификатору в элементе управления представления списка. Вы можете использовать этот макрос или отправить сообщение LVM_SORTGROUPS явным образом.

[ListView_SortItems](#)

Использует определяемую приложением функцию сравнения для сортировки элементов элемента управления представлением списка. Индекс каждого элемента изменяется в соответствии с новой последовательностью. Вы можете использовать этот макрос или отправить сообщение LVM_SORTITEMS явным образом.

[ListView_SortItemsEx](#)

Использует определяемую приложением функцию сравнения для сортировки элементов элемента управления представлением списка. Индекс каждого элемента изменяется в соответствии с новой последовательностью. Вы можете использовать этот макрос или отправить сообщение LVM_SORTITEMSEX явным образом.

[ListView_SubItemHitTest](#)

Определяет, какой элемент представления списка или подэлемент находится в заданной позиции. Вы можете использовать этот макрос или отправить сообщение LVM_SUBITEMHITTEST явным образом. (ListView_SubItemHitTest)

[ListView_SubItemHitTestEx](#)

Определяет, какой элемент представления списка или подэлемент находится в заданной позиции. Вы можете использовать этот макрос или отправить сообщение LVM_SUBITEMHITTEST явным образом. (ListView_SubItemHitTestEx)

[ListView_Update](#)

Обновления элемент представления списка. Если элемент управления представлением списка имеет стиль LVS_AUTOARRANGE, этот макрос приводит к упорядочению элемента управления представлением списка. Вы можете использовать этот макрос или отправить сообщение LVM_UPDATE явным образом.

[LoadIconMetric](#)

Загружает указанный ресурс значка с системной метрикой, указанной клиентом.

[LoadIconWithScaleDown](#)

Загружает значок. Если значок не имеет стандартного размера, эта функция масштабирует изображение большего размера, а не масштабирует изображение меньшего размера.

[MakeDragList](#)

Изменяет указанный список с одним выделением на список перетаскивания.

[MAKEIPADDRESS](#)

Упаковывает четыре байтовых значения в один LPARAM, подходящий для использования с сообщением IPM_SETADDRESS.

[MAKEIPRANGE](#)

Упаковывает два байтовых значения в один LPARAM, подходящий для использования с сообщением IPM_SETRANGE.

[MenuHelp](#)

Обрабатывает WM_MENUSELECT и WM_COMMAND сообщения и отображает текст справки о текущем меню в указанном окне состояния.

[MonthCal_GetCalendarBorder](#)

Возвращает размер границы (в пикселях) элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_GETCALENDARBORDER явным образом.

[MonthCal_GetCalendarCount](#)

Возвращает количество календарей, отображаемых в данный момент в элементе управления "Календарь". Вы можете использовать этот макрос или отправить сообщение MCM_GETCALENDARCOUNT явным образом.

[MonthCal_GetCalendarGridInfo](#)

Возвращает сведения о сетке календаря.

[MonthCal_GetCALID](#)

Возвращает идентификатор текущего календаря для заданного элемента управления календарем. Вы можете использовать этот макрос или отправить сообщение MCM_GETCALID явным образом.

[MonthCal_GetColor](#)

Извлекает цвет для заданной части элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_GETCOLOR явным образом.

[MonthCal_GetCurrentView](#)

Получает представление для элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_GETCURRENTVIEW явным образом.

[MonthCal_GetCurSel](#)

Извлекает текущую выбранную дату. Вы можете использовать этот макрос или отправить сообщение MCM_GETCURSEL явным образом.

[MonthCal_GetFirstDayOfWeek](#)

Извлекает первый день недели для элемента управления "Календарь месяца". Вы можете использовать этот макрос или отправить сообщение MCM_GETFIRSTDAYOFWEEK явным образом.

[MonthCal_GetMaxSelCount](#)

Извлекает максимальный диапазон дат, который можно выбрать в элементе управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_GETMAXSELCOUNT явным образом.

[MonthCal_GetMaxTodayWidth](#)

Извлекает максимальную ширину строки "сегодня" в элементе управления "Календарь на месяц". Сюда входят текст метки и текст даты. Вы можете использовать этот макрос или отправить сообщение MCM_GETMAXTODAYWIDTH явным образом.

[MonthCal_GetMinReqRect](#)

Возвращает минимальный размер, необходимый для отображения полного месяца в элементе управления "Календарь месяца". Сведения о размере представлены в виде структуры RECT. Вы можете использовать этот макрос или отправить сообщение MCM_GETMINREQRECT явным образом.

[MonthCal_GetMonthDelta](#)

Получает скорость прокрутки для элемента управления "Календарь на месяц". Скорость прокрутки — это количество месяцев, в течение которых элемент управления перемещает свой дисплей, когда пользователь нажимает кнопку прокрутки. Вы можете использовать этот макрос или отправить сообщение MCM_GETMONTHDELTA явным образом.

[MonthCal_GetMonthRange](#)

Извлекает сведения о дате (с помощью структур SYSTEMTIME), которые представляют высокие и низкие пределы отображения элемента управления календарем на месяц. Вы можете использовать этот макрос или отправить сообщение MCM_GETMONTHRANGE явным образом.

[MonthCal_GetRange](#)

Извлекает минимальные и максимальные допустимые даты, установленные для элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_GETRANGE явным образом.

[MonthCal_GetSelRange](#)

Извлекает сведения о дате, представляющие верхний и нижний пределы диапазона дат, выбранного пользователем. Вы можете использовать этот макрос или отправить сообщение MCM_GETSEL RANGE явным образом.

[MonthCal_GetToday](#)

Извлекает сведения о дате, указанной как "сегодня" для элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_GETTODAY явным образом.

[MonthCal_GetUnicodeFormat](#)

Извлекает флаг символьного формата Юникода для элемента управления . Вы можете использовать этот макрос или отправить сообщение MCM_GETUNICODEFORMAT явным образом.

[MonthCal_HitTest](#)

Определяет, какая часть элемента управления "Календарь на месяц" находится в заданной точке экрана. Вы можете использовать этот макрос или отправить сообщение MCM_HITTEST явным образом.

[MonthCal_SetCalendarBorder](#)

Задает размер границы (в пикселях) элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_SETCALENDAR BORDER явным образом.

[MonthCal_SetCALID](#)

Задает идентификатор календаря для заданного элемента управления календарем. Вы можете использовать этот макрос или отправить сообщение MCM_SETCALID явным образом.

[MonthCal_SetColor](#)

Задает цвет для заданной части элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_SET COLOR явным образом.

[MonthCal_SetCurrentView](#)

Задает представление для элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_SETCURRENTVIEW явным образом.

[MonthCal_SetCurSel](#)

Задает текущую дату, выбранную для элемента управления "Календарь на месяц". Если указанная дата не отображается, элемент управления обновляет отображение, чтобы отобразить его. Вы можете использовать этот макрос или отправить сообщение MCM_SETCURSEL явным образом.

[MonthCal_SetDayState](#)

Задает состояния дня для всех месяцев, которые отображаются в элементе управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_SETDAYSTATE явным образом.

[MonthCal_SetFirstDayOfWeek](#)

Задает первый день недели для элемента управления "Календарь месяца". Вы можете использовать этот макрос или отправить сообщение MCM_SETFIRSTDAYOFWEEK явным образом.

[MonthCal_SetMaxSelCount](#)

Задает максимальное количество дней, которое можно выбрать в элементе управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_SETMAXSELCOUNT явным образом.

[MonthCal_SetMonthDelta](#)

Задает частоту прокрутки для элемента управления "Календарь на месяц". Скорость прокрутки — это количество месяцев, в течение которых элемент управления перемещает свой дисплей, когда пользователь нажимает кнопку прокрутки. Вы можете использовать этот макрос или отправить сообщение MCM_SETMONTHDELTA явным образом.

[MonthCal_SetRange](#)

Задает минимальные и максимально допустимые даты для элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_SETRANGE явным образом.

[MonthCal_SetSelRange](#)

Задает для элемента управления календарь на месяц заданный диапазон дат. Вы можете использовать этот макрос или отправить сообщение MCM_SETSEL RANGE явным образом.

[MonthCal_SetToday](#)

Задает выбор "сегодня" для элемента управления "Календарь на месяц". Вы можете использовать этот макрос или отправить сообщение MCM_SETTODAY явным образом.

[MonthCal_SetUnicodeFormat](#)

Задает флаг символьного формата Юникода для элемента управления .
(MonthCal_SetUnicodeFormat)

[MonthCal_SizeRectToMin](#)

Вычисляет, сколько календарей поместится в заданном прямоугольнике, а затем возвращает минимальный размер прямоугольника, который должен соответствовать такому количеству календарей. Вы можете использовать этот макрос или отправить сообщение MCM_SIZERECTTOMIN явным образом.

[Pager_ForwardMouse](#)

Включает или отключает переадресацию мыши для элемента управления пейджером. Если переадресация мыши включена, элемент управления пейджера перенаправит WM_MOUSEMOVE сообщения в автономное окно. Вы можете использовать этот макрос или отправить сообщение PGM_FORWARDMOUSE явным образом.

[Pager_GetBkColor](#)

Извлекает текущий цвет фона для элемента управления пейджером. Вы можете использовать этот макрос или отправить сообщение PGM_GETBKCOLOR явным образом.

[Pager_GetBorder](#)

Извлекает текущий размер границы для элемента управления пейджера. Вы можете использовать этот макрос или отправить сообщение PGM_GETBORDER явным образом.

[Pager_GetButtonSize](#)

Извлекает текущий размер кнопки для элемента управления пейджера. Вы можете использовать этот макрос или отправить сообщение PGM_GETBUTTONSIZE явным образом.

[Pager_GetButtonState](#)

Извлекает состояние указанной кнопки в элементе управления пейджера. Вы можете использовать этот макрос или отправить сообщение PGM_GETBUTTONSTATE явным образом.

[Pager_GetDropTarget](#)

Извлекает указатель интерфейса IDropTarget элемента управления страничного навиджа. Вы можете использовать этот макрос или отправить сообщение PGM_GETDROPTARGET явным образом.

[Pager_GetPos](#)

Извлекает текущую позицию прокрутки элемента управления пейджера. Вы можете использовать этот макрос или отправить сообщение PGM_GETPOS явным образом.

[Pager_RecalcSize](#)

Заставляет элемент управления пейджера пересчитывать размер автономного окна. Использование этого макроса приведет к отправке PGN_CALCSIZE уведомления. Вы можете использовать этот макрос или отправить сообщение PGM_RECALCSIZE явным образом.

[Pager_SetBkColor](#)

Задает текущий цвет фона для элемента управления пейджером. Вы можете использовать этот макрос или отправить сообщение PGM_SETBKCOLOR явным образом.

[Pager_SetBorder](#)

Задает текущий размер границы для элемента управления пейджера. Вы можете использовать этот макрос или отправить сообщение PGM_SETBORDER явным образом.

[Pager_SetButtonSize](#)

Задает текущий размер кнопки для элемента управления пейджером. Вы можете использовать этот макрос или отправить сообщение PGM_SETBUTTONSIZE явным образом.

[Pager_SetChild](#)

Задает автономное окно для элемента управления пейджером.

[Pager_SetPos](#)

Задает положение прокрутки для элемента управления страничного навиджа. Вы можете использовать этот макрос или отправить сообщение PGM_SETPOS явным образом.

[Pager_SetScrollInfo](#)

Задает параметры прокрутки элемента управления пейджера, включая значение времени ожидания, количество строк времени ожидания и пиксели на строку. Вы можете использовать этот макрос или отправить сообщение PGM_SETSETSCROLLINFO явным образом.

[RemoveWindowSubclass](#)

Удаляет обратный вызов подкласса из окна.

[SECOND_IPADDRESS](#)

Извлекает значение поля 1 из упакованного IP-адреса, полученного с сообщением IPM_GETADDRESS.

[SetWindowSubclass](#)

Устанавливает или обновляет обратный вызов подкласса окна.

[ShowHideMenuCtl](#)

Задает или удаляет атрибут проверка метки указанного пункта меню и отображает или скрывает соответствующий элемент управления.

[TabCtrl_AdjustRect](#)

Вычисляет область отображения элемента управления табуляции с учетом прямоугольника окна или прямоугольник окна, соответствующий заданной области отображения. Вы можете использовать этот макрос или отправить сообщение TCM_ADJUSTRECT явным образом.

[TabCtrl_DeleteAllItems](#)

Удаляет все элементы из элемента управления вкладки. Вы можете использовать этот макрос или отправить сообщение TCM_DELETEALLITEMS явным образом.

[TabCtrl_DeleteItem](#)

Удаляет элемент из элемента управления вкладки. Вы можете использовать этот макрос или отправить сообщение TCM_DELETEITEM явным образом.

[TabCtrl_DeselectAll](#)

Сбрасывает элементы в элементе управления "Вкладка", очищая все элементы, для TCIS_BUTTONPRESSED состояния. Вы можете использовать этот макрос или отправить сообщение TCM_DESELECTALL явным образом.

[TabCtrl_GetCurFocus](#)

Возвращает индекс элемента, который имеет фокус в элементе управления tab. Вы можете использовать этот макрос или отправить сообщение TCM_GETCURFOCUS явным образом.

[TabCtrl_GetCurSel](#)

Определяет текущую выбранную вкладку в элементе управления вкладками. Вы можете использовать этот макрос или отправить сообщение TCM_GETCURSEL явным образом.

[TabCtrl_GetExtendedStyle](#)

Извлекает расширенные стили, которые в настоящее время используются для элемента управления вкладкой. Вы можете использовать этот макрос или отправить сообщение TCM_GETEXTENDEDSTYLE явным образом.

[TabCtrl_GetImageList](#)

Извлекает список изображений, связанный с элементом управления вкладкой. Вы можете использовать этот макрос или отправить сообщение TCM_GETIMAGELIST явным образом.

[TabCtrl_GetItem](#)

Извлекает сведения о вкладке в элементе управления "Вкладка". Вы можете использовать этот макрос или отправить сообщение TCM_GETITEM явным образом.

[TabCtrl_GetItemCount](#)

Извлекает число вкладок в наборе вкладок. Вы можете использовать этот макрос или отправить сообщение TCM_GETITEMCOUNT явным образом.

[TabCtrl_GetItemRect](#)

Извлекает ограничивающий прямоугольник для вкладки в элементе управления tab. Вы можете использовать этот макрос или отправить сообщение TCM_GETITEMRECT явным образом.

[TabCtrl_GetRowCount](#)

Извлекает текущее количество строк вкладок в элементе управления вкладками. Вы можете использовать этот макрос или отправить сообщение TCM_GETROWCOUNT явным образом.

[TabCtrl_GetToolTips](#)

Извлекает дескриптор элемента управления подсказкой, связанного с элементом управления tab. Вы можете использовать этот макрос или отправить сообщение TCM_GETTOOLTIPS явным образом.

[TabCtrl_GetUnicodeFormat](#)

Извлекает флаг символьного формата ЮНИКОДа для элемента управления . Вы можете использовать этот макрос или отправить сообщение TCM_GETUNICODEFORMAT явным образом.

[TabCtrl_HighlightItem](#)

Задает состояние выделения элемента вкладки. Вы можете использовать этот макрос или отправить сообщение TCM_HIGHLIGHTITEM явным образом.

[TabCtrl_HitTest](#)

Определяет, какая вкладка (при наличии) находится в указанной позиции экрана. Вы можете использовать этот макрос или отправить сообщение TCM_HITTEST явным образом.

[TabCtrl_InsertItem](#)

Вставляет новую вкладку в элемент управления вкладкой. Вы можете использовать этот макрос или отправить сообщение TCM_INSERTITEM явным образом.

[TabCtrl_RemoveImage](#)

Удаляет изображение из списка изображений элемента управления вкладкой. Вы можете использовать этот макрос или отправить сообщение TCM_REMOVEIMAGE явным образом.

[TabCtrl_SetCurFocus](#)

Устанавливает фокус на указанную вкладку в элементе управления вкладкой. Вы можете использовать этот макрос или отправить сообщение TCM_SETCURFOCUS явным образом.

[TabCtrl_SetCurSel](#)

Выбирает вкладку в элементе управления "Вкладка". Вы можете использовать этот макрос или отправить сообщение TCM_SETCURSEL явным образом.

[TabCtrl_SetExtendedStyle](#)

Задает расширенные стили, которые будет использовать элемент управления вкладкой. Вы можете использовать этот макрос или отправить сообщение TCM_SETEXTENDEDSTYLE явным образом.

[TabCtrl_SetImageList](#)

Назначает список изображений элементу управления вкладкой. Вы можете использовать этот макрос или отправить сообщение TCM_SETIMAGELIST явным образом.

[TabCtrl_SetItem](#)

Задает некоторые или все атрибуты вкладки. Вы можете использовать этот макрос или отправить сообщение TCM_SETITEM явным образом.

[TabCtrl_SetItemExtra](#)

Задает количество байтов на вкладку, зарезервированное для определяемых приложением данных в элементе управления вкладками. Вы можете использовать этот макрос или отправить сообщение TCM_SETITEMEXTRA явным образом.

[TabCtrl_SetItemSize](#)

Задает ширину и высоту вкладок в элементе управления вкладки фиксированной ширины или нарисованной владельцем. Вы можете использовать этот макрос или отправить сообщение TCM_SETITEMSIZE явным образом.

[TabCtrl_SetMinTabWidth](#)

Задает минимальную ширину элементов в элементе управления "Вкладка". Вы можете использовать этот макрос или отправить сообщение TCM_SETMINTABWIDTH явным образом.

[TabCtrl_SetPadding](#)

Задает объем пространства (заполнение) вокруг значка и метки каждой вкладки в элементе управления вкладки. Вы можете использовать этот макрос или отправить сообщение TCM_SETPADDING явным образом.

[TabCtrl_SetToolTips](#)

Назначает элемент управления подсказкой элементу управления "Вкладка". Вы можете использовать этот макрос или отправить сообщение TCM_SETTOOLTIPS явным образом.

[TabCtrl_SetUnicodeFormat](#)

Задает флаг символьного формата Юникода для элемента управления .
(TabCtrl_SetUnicodeFormat)

[TaskDialog](#)

Функция TaskDialog создает, отображает и управляет диалоговым окном задачи.

[TaskDialogIndirect](#)

Функция TaskDialogIndirect создает, отображает и управляет диалоговым окном задачи.

[THIRD_IPADDRESS](#)

Извлекает значение поля 2 из упакованного IP-адреса, полученного с сообщением IPM_GETADDRESS.

[TreeView_CreateDragImage](#)

Создает перетаскивание растрового рисунка для указанного элемента в элементе управления в виде дерева.

[TreeView_DeleteAllItems](#)

Удаляет все элементы из элемента управления в виде дерева.

[TreeView_DeleteItem](#)

Удаляет элемент и все его дочерние элементы из элемента управления в виде дерева. Вы также можете отправить TVM_DELETEITEM сообщение явным образом.

[TreeView_EditLabel](#)

Начинает редактирование текста указанного элемента на месте, заменяя его на односторонний элемент управления редактирования, содержащий текст.

[TreeView_EndEditLabelNow](#)

Завершает редактирование метки элемента представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_ENDEDITLABELNOW явным образом.

[TreeView_EnsureVisible](#)

Обеспечивает видимость элемента представления в виде дерева, разворачивая родительский элемент или при необходимости прокручивая элемент управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_ENSUREVISIBLE явным образом.

[TreeView_Expand](#)

Макрос TreeView_Expand расширяет или сворачивает список дочерних элементов, связанных с указанным родительским элементом, если таковые есть. Вы можете использовать этот макрос или отправить сообщение TVM_EXPAND явным образом.

[TreeView_GetBkColor](#)

Извлекает текущий цвет фона элемента управления. Вы можете использовать этот макрос или отправить сообщение TVM_GETBKCOLOR явным образом.

[TreeView_GetCheckState](#)

Возвращает состояние проверка указанного элемента. Вы также можете использовать TVM_GETITEMSTATE сообщение напрямую.

[TreeView_GetChild](#)

Извлекает первый дочерний элемент указанного элемента представления в виде дерева. Вы можете использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_CHILD.

[TreeView_GetCount](#)

Извлекает количество элементов в элементе управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETCOUNT явным образом.

[TreeView_GetDropHilight](#)

Извлекает элемент представления в виде дерева, который является целевым объектом операции перетаскивания. Вы можете использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_DROPHILITE.

[TreeView_GetEditControl](#)

Извлекает дескриптор элемента управления редактированием, используемого для изменения текста элемента представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETEDITCONTROL явным образом.

[TreeView_GetExtendedStyle](#)

Извлекает расширенный стиль для указанного элемента управления в виде дерева. Используйте этот макрос или отправьте сообщение TVM_GETEXTENDEDSTYLE явным образом.

[TreeView_GetFirstVisible](#)

Извлекает первый видимый элемент в окне элемента управления в виде дерева. Можно использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_FIRSTVISIBLE.

[TreeView_GetImageList](#)

Извлекает дескриптор в обычный список изображений или изображений состояния, связанный с элементом управления представлением в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETIMAGELIST явным образом.

[TreeView_GetIndent](#)

Извлекает количество (в пикселях) отступов дочерних элементов относительно их родительских элементов. Вы можете использовать этот макрос или отправить сообщение TVM_GETINDENT явным образом.

[TreeView_GetInsertMarkColor](#)

Извлекает цвет, используемый для рисования метки вставки для представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETINSERTMARKCOLOR явным образом.

[TreeView_GetSearchString](#)

Извлекает строку добавочного поиска для элемента управления в виде дерева. Элемент управления в виде дерева использует строку добавочного поиска для выбора элемента на основе символов, введенных пользователем. Вы можете использовать этот макрос или отправить сообщение TVM_GETSEARCHSTRING явным образом.

[TreeView_GetItem](#)

Извлекает некоторые или все атрибуты элемента представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETITEM явным образом.

[TreeView_GetItemHeight](#)

Извлекает текущую высоту элементов представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETITEMHEIGHT явным образом.

[TreeView_GetItemPartRect](#)

Извлекает максимально возможный ограничивающий прямоугольник, который представляет собой "зону попадания" для указанной части элемента. Используйте этот макрос или отправьте сообщение TVM_GETITEMPARTRECT явным образом.

[TreeView_GetItemRect](#)

Извлекает ограничивающий прямоугольник для элемента представления в виде дерева и указывает, является ли элемент видимым. Вы можете использовать этот макрос или отправить сообщение TVM_GETITEMRECT явным образом.

[TreeView_GetItemState](#)

Извлекает некоторые или все атрибуты состояния элемента представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETITEMSTATE явным образом.

[TreeView_GetLastVisible](#)

Извлекает последний развернутый элемент в элементе управления в виде дерева. При этом не извлекается последний элемент, видимый в окне представления в виде дерева. Можно использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_LASTVISIBLE.

[TreeView_GetLineColor](#)

Возвращает текущий цвет линии. Вы также можете использовать сообщение TVM_GETLINECOLOR напрямую.

[TreeView_GetNextItem](#)

Извлекает элемент представления в виде дерева, который имеет указанную связь с указанным элементом. Вы можете использовать этот макрос, использовать один из макросов TreeView_Get, описанных ниже, или отправить сообщение TVM_GETNEXTITEM явным образом.

[TreeView_GetNextSelected](#)

Извлекает элемент представления в виде дерева, который имеет связь TVGN_NEXTSELECTED с указанным элементом дерева.

[TreeView_GetNextSibling](#)

Извлекает следующий одноуровневый элемент указанного элемента в элементе управления в виде дерева. Вы можете использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_NEXT.

[TreeView_GetNextVisible](#)

Извлекает следующий видимый элемент, следующий за указанным элементом в элементе управления в виде дерева. Можно использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_NEXTVISIBLE.

[TreeView_GetParent](#)

Извлекает родительский элемент указанного элемента представления в виде дерева. Вы можете использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_PARENT.

[TreeView_GetPrevSibling](#)

Извлекает предыдущий элемент того же уровня указанного элемента в элементе управления в виде дерева. Вы можете использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_PREVIOUS.

[TreeView_GetPrevVisible](#)

Извлекает первый видимый элемент, который предшествует указанному элементу в элементе управления в виде дерева. Вы можете использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_PREVIOUSVISIBLE.

[TreeView_GetRoot](#)

Извлекает самый верхний или самый первый элемент элемента управления в виде дерева. Можно использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_ROOT.

[TreeView_GetScrollTime](#)

Извлекает максимальное время прокрутки элемента управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETSCROLLTIME явным образом.

[TreeView_GetSelectedCount](#)

макрос TreeView_GetSelectedCount

[TreeView_GetSelection](#)

Извлекает текущий выбранный элемент в элементе управления в виде дерева. Вы можете использовать этот макрос или явно отправить сообщение TVM_GETNEXTITEM с флагом TVGN_CARET.

[TreeView_GetTextColor](#)

Извлекает текущий цвет текста элемента управления . Вы можете использовать этот макрос или отправить сообщение TVM_GETTEXTCOLOR явным образом.

[TreeView_GetToolTips](#)

Извлекает дескриптор дочернего элемента управления tooltip, используемого элементом управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETTOOLTIPS явным образом.

[TreeView_GetUnicodeFormat](#)

Извлекает флаг символьного формата Юникода для элемента управления . Вы можете использовать этот макрос или отправить сообщение TVM_GETUNICODEFORMAT явным образом.

[TreeView_GetVisibleCount](#)

Получает количество элементов, которые могут быть полностью видны в клиентском окне элемента управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_GETVISIBLECOUNT явным образом.

[TreeView_HitTest](#)

Определяет расположение указанной точки относительно клиентской области элемента управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_HITTEST явным образом.

[TreeView_InsertItem](#)

Вставляет новый элемент в элемент управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_INSERTITEM явным образом.

[TreeView_MapAccIDToHTREEITEM](#)

Сопоставляет идентификатор специальных возможностей с HTREEITEM. Вы можете использовать этот макрос или отправить сообщение TVM_MAPACCIDTOHTREEITEM явным образом.

[TreeView_MapHTREEITEMToAccID](#)

Сопоставляет HTREEITEM с идентификатором специальных возможностей. Вы можете использовать этот макрос или отправить сообщение TVM_MAPHTREEITEMTOACCID явным образом.

[TreeView_Select](#)

Выбирает указанный элемент в виде дерева, прокручивает его в представление или перерисовывает элемент в стиле, используемом для указания целевого объекта операции перетаскивания.

[TreeView_SelectDropTarget](#)

Перерисовывает указанный элемент управления в виде дерева в стиле, используемом для указания целевого объекта операции перетаскивания. Можно использовать этот макрос или макрос TreeView_Select либо отправить сообщение TVM_SELECTITEM явным образом.

[TreeView_SelectItem](#)

Выбирает указанный элемент представления в виде дерева. Можно использовать этот макрос или макрос TreeView_Select либо отправить сообщение TVM_SELECTITEM явным образом.

[TreeView_SelectSetFirstVisible](#)

Прокручивает элемент управления в виде дерева по вертикали, чтобы убедиться, что указанный элемент является видимым.

[TreeView_SetAutoScrollInfo](#)

Задает сведения, используемые для определения характеристик автоматической прокрутки. Используйте этот макрос или отправьте сообщение TVM_SETAUTOSCROLLINFO явным образом.

[TreeView_SetBkColor](#)

Задает цвет фона элемента управления. Вы можете использовать этот макрос или отправить сообщение TVM_SETBKCOLOR явным образом.

[TreeView_SetBorder](#)

Задает размер границы для элементов в элементе управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETBORDER явным образом.

[TreeView_SetCheckState](#)

Устанавливает для изображения состояния элемента значение "checked" или "unchecked". Вы также можете использовать сообщение TVM_SETITEM напрямую.

[TreeView_SetExtendedStyle](#)

Задает расширенный стиль для указанного элемента управления TreeView. Используйте этот макрос или отправьте сообщение TVM_SETEXTENDEDSTYLE явным образом.

[TreeView_SetHot](#)

Задает горячий элемент для элемента управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETHOT явным образом.

[TreeView_SetImageList](#)

Задает обычный список изображений или изображений состояния для элемента управления в виде дерева и перерисовывает элемент управления с помощью новых изображений. Вы можете использовать этот макрос или отправить сообщение TVM_SETIMAGELIST явным образом.

[TreeView_SetIndent](#)

Задает ширину отступа для элемента управления в виде дерева и перерисовывает элемент управления в соответствии с новой шириной. Вы можете использовать этот макрос или отправить сообщение TVM_SETINDENT явным образом.

[TreeView_SetInsertMark](#)

Задает метку вставки в элементе управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETINSERTMARK явным образом.

[TreeView_SetInsertMarkColor](#)

Задает цвет, используемый для рисования метки вставки для представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETINSERTMARKCOLOR явным образом.

[TreeView_SetItem](#)

Макрос TreeView_SetItem задает некоторые или все атрибуты элемента представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETITEM явным образом.

[TreeView_SetItemHeight](#)

Задает высоту элементов представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETITEMHEIGHT явным образом.

[TreeView_SetItemState](#)

Задает атрибуты состояния элемента представления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETITEM явным образом.

[TreeView_SetLineColor](#)

Задает текущий цвет линии. Вы также можете использовать сообщение TVM_SETLINECOLOR напрямую.

[TreeView_SetScrollTime](#)

Задает максимальное время прокрутки для элемента управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETSCROLLTIME явным образом.

[TreeView_SetTextColor](#)

Задает цвет текста элемента управления. Вы можете использовать этот макрос или отправить сообщение TVM_SETTEXTCOLOR явным образом.

[TreeView_SetToolTips](#)

Задает дочерний элемент управления "Подсказка" в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SETTOOLTIPS явным образом.

[TreeView_SetUnicodeFormat](#)

Задает флаг символьного формата Юникода для элемента управления .
(TreeView_SetUnicodeFormat)

[TreeView_ShowInfoTip](#)

Отображает подсказку для указанного элемента в элементе управления в виде дерева. Используйте этот макрос или отправьте сообщение TVM_SHOWINFOTIP явным образом.

[TreeView_SortChildren](#)

Сортирует дочерние элементы указанного родительского элемента в элементе управления в виде дерева. Вы можете использовать этот макрос или отправить сообщение TVM_SORTCHILDREN явным образом.

[TreeView_SortChildrenCB](#)

Сортирует элементы представления в виде дерева с помощью определяемой приложением функции обратного вызова, которая сравнивает элементы. Вы можете использовать этот макрос или отправить сообщение TVM_SORTCHILDRENCB явным образом.

[UninitializeFlatSB](#)

Неинициализирует плоские полосы прокрутки для определенного окна. Указанное окно будет отменить изменения к стандартным полосам прокрутки.

Функции обратного вызова

[PFNLVGROUPCOMPARE](#)

Функция LVGroupCompare — это определяемая приложением функция обратного вызова, используемая с LVM_INSERTGROUPSORTED и LVM_SORTGROUPS сообщениями.

[PFTASKDIALOGCALLBACK](#)

Функция TaskDialogCallbackProc — это определяемая приложением функция, используемая с функцией TaskDialogIndirect.

[ПОДКЛАССПРОЦ](#)

Определяет прототип функции обратного вызова, используемой RemoveWindowSubclass и SetWindowSubclass.

Структуры

[BUTTON_IMAGELIST](#)

Содержит сведения о списке изображений, используемых с элементом управления "Кнопка".

BUTTON_SPLITINFO

Содержит сведения, определяющие разделенную кнопку (стили BS_SPLITBUTTON и BS_DEFSPLITBUTTON). Используется с сообщениями BCM_GETSPLITINFO и BCM_SETSPLITINFO.

COLORMAP

Содержит сведения, используемые функцией CreateMappedBitmap для сопоставления цветов растрового рисунка.

COLORSCHEME

Содержит сведения для рисования кнопок на панели инструментов или на панели инструментов.

COMBOBOXEXITEMA

Содержит сведения об элементе в элементе управления ComboBoxEx. (ANSI)

COMBOBOXEXITEMW

Содержит сведения об элементе в элементе управления ComboBoxEx. (Юникод)

DATETIMEPICKERINFO

Содержит сведения об элементе управления выбора даты и времени (DTP).

DRAGLISTINFO

Содержит сведения о событии перетаскивания. Указатель на DRAGLISTINFO передается в качестве параметра lParam сообщения списка перетаскивания.

EDITBALLOONTIP

Содержит сведения о наконечнике выноски, связанной с элементом управления "Кнопка".

HD_TEXTFILTERA

Содержит сведения о текстовых фильтрах элементов управления заголовками. (ANSI)

HD_TEXTFILTERW

Содержит сведения о текстовых фильтрах элементов управления заголовками. (Юникод)

HDHITTESTINFO

Содержит сведения о проверке нажатия. Эта структура используется с сообщением HDM_HITTEST и заменяет структуру HD_HITTESTINFO.

[HDITEMA](#)

Содержит сведения об элементе в элементе управления заголовком. Эта структура заменяет структуру HD_ITEM. (ANSI)

[HDITEMW](#)

Содержит сведения об элементе в элементе управления заголовком. Эта структура заменяет структуру HD_ITEM. (Юникод)

[HDLAYOUT](#)

Содержит сведения, используемые для задания размера и положения элемента управления заголовком. HD_LAYOUT используется с сообщением HDM_LAYOUT. Эта структура заменяет структуру HD_LAYOUT.

[IMAGEINFO](#)

Структура IMAGEINFO содержит сведения об изображении в списке изображений. Эта структура используется с функцией `IImageList::GetImageInfo`.

[IMAGELISTDRAWPARAMS](#)

Структура IMAGELISTDRAWPARAMS содержит сведения об операции рисования списка изображений и используется с функцией `IImageList::Draw`.

[INITCOMMONCONTROLSEX](#)

Содержит сведения, используемые для загрузки общих классов элементов управления из библиотеки динамической компоновки (DLL). Эта структура используется с функцией `InitCommonControlsEx`.

[LHITTESTINFO](#)

Используется для получения сведений о ссылке, соответствующей заданному расположению.

[LITEM](#)

Используется для задания и получения сведений об элементе ссылки.

[LVBKIMAGEA](#)

Содержит сведения о фоновом изображении элемента управления "Представление списка". Эта структура используется как для задания, так и для получения сведений о фоновом изображении. (ANSI)

[LVBKIMAGEW](#)

Содержит сведения о фоновом изображении элемента управления "Представление списка". Эта структура используется как для задания, так и для получения сведений о фоновом изображении. (Юникод)

[LVCOLUMNNA](#)

Содержит сведения о столбце в представлении отчета. Эта структура используется как для создания столбцов, так и для управления ими. Эта структура заменяет структуру LV_COLUMN. (ANSI)

[LVCOLUMNW](#)

Содержит сведения о столбце в представлении отчета. Эта структура используется как для создания столбцов, так и для управления ими. Эта структура заменяет структуру LV_COLUMN. (Юникод)

[LVFINDINFOA](#)

Содержит сведения, используемые при поиске элемента представления списка. Эта структура идентична LV_FINDINFO но была переименована в соответствии со стандартными соглашениями об именовании. (ANSI)

[LVFINDINFOW](#)

Содержит сведения, используемые при поиске элемента представления списка. Эта структура идентична LV_FINDINFO но была переименована в соответствии со стандартными соглашениями об именовании. (Юникод)

[LVFOOTERINFO](#)

Содержит сведения о нижнем колонтитуле в элементе управления "Представление списка".

[LVFOOTERITEM](#)

Содержит сведения о элементе нижнего колонтитула.

[LVGROUP](#)

Используется для задания и извлечения групп.

[LVGROUPMETRICS](#)

Содержит сведения о отображении групп в элементе управления "Представление списка".

[LVHITTESTINFO](#)

Содержит сведения о проверке нажатия.

[LVINSERTGROUPSORTED](#)

Используется для сортировки групп. Используется с LVM_INSERTGROUPSORTED.

[LVINSERTMARK](#)

Используется для описания точек вставки.

[LVITEMA](#)

Задает или получает атрибуты элемента представления списка. Эта структура была обновлена для поддержки нового значения маски (LVIF_INDENT), которое включает отступ элемента. Эта структура заменяет структуру LV_ITEM. (ANSI)

[LVITEMINDEX](#)

Содержит сведения об индексе элемента представления списка.

[LVITEMW](#)

Задает или получает атрибуты элемента представления списка. Эта структура была обновлена для поддержки нового значения маски (LVIF_INDENT), которое включает отступ элемента. Эта структура заменяет структуру LV_ITEM. (Юникод)

[LVSETINFOTIP](#)

Предоставляет сведения о тексте подсказки, который необходимо задать.

[LVTILEINFO](#)

Предоставляет сведения об элементе в элементе управления в представлении списка при его отображении в представлении плитки.

[LVTILEVIEWINFO](#)

Предоставляет сведения об элементе управления представлением списка при его отображении в представлении плитки.

[MCGRIDINFO](#)

Содержит сведения о части элемента управления "Календарь".

[MCHITTESTINFO](#)

Содержит сведения, относящиеся к точкам проверки попадания для элемента управления календарем на месяц. Эта структура используется с сообщением MCM_HITTEST и соответствующим макросом MonthCal_HitTest.

NMBCDROPDOWN

Содержит сведения об уведомлении BCN_DROPDOWN.

NMBCHOTITEM

Содержит сведения о перемещении указателя мыши на элемент управления "Кнопка".

NMCBEDRAGBEGINA

Содержит сведения, используемые с кодом уведомления CBEN_DRAGBEGIN. (ANSI)

NMCBEDRAGBEGINW

Содержит сведения, используемые с кодом уведомления CBEN_DRAGBEGIN. (Юникод)

NMCBEENDEDITA

Содержит сведения о завершении операции редактирования в элементе управления ComboBoxEx. Эта структура используется с кодом уведомления CBEN_ENDEDIT. (ANSI)

NMCBEENDEDITW

Содержит сведения о завершении операции редактирования в элементе управления ComboBoxEx. Эта структура используется с кодом уведомления CBEN_ENDEDIT. (Юникод)

NMCHAR

Содержит сведения, используемые с символьными уведомлениями.

NMCOMBOBOXEXA

Содержит сведения, относящиеся к элементам ComboBoxEx для использования с кодами уведомлений. (ANSI)

NMCOMBOBOXEXW

Содержит сведения, относящиеся к элементам ComboBoxEx для использования с кодами уведомлений. (Юникод)

NMCUSTOMDRAW

Содержит сведения, относящиеся к коду уведомления NM_CUSTOMDRAW.

NMCUSTOMSPLITRECTINFO

Содержит сведения о двух прямоугольниках разделенной кнопки. Отправлено с уведомлением NM_GETCUSTOMSPLITRECT.

NMCUSTOMTEXT

Содержит сведения, используемые с пользовательским текстовым уведомлением.

NMDATETIMECHANGE

Содержит сведения об изменении, которое произошло в элементе управления выбора даты и времени (DTP). Эта структура используется с кодом уведомления DTN_DATETIMECHANGE.

NMDATETIMEFORMATA

Содержит сведения о части строки формата, определяющей поле обратного вызова в элементе управления "Выбор даты и времени" (DTP). (ANSI)

NMDATETIMEFORMATQUERYA

Содержит сведения о поле обратного вызова средства выбора даты и времени (DTP). (ANSI)

NMDATETIMEFORMATQUERYW

Содержит сведения о поле обратного вызова средства выбора даты и времени (DTP). (Юникод)

NMDATETIMEFORMATW

Содержит сведения о части строки формата, определяющей поле обратного вызова в элементе управления "Выбор даты и времени" (DTP). (Юникод)

NMDATETIMESTRINGA

Содержит сведения, относящиеся к операции редактирования, которая выполнялась в элементе управления "Выбор даты и времени" (DTP). Это сообщение используется с кодом уведомления DTN_USERSTRING. (ANSI)

NMDATETIMESTRINGW

Содержит сведения, относящиеся к операции редактирования, которая выполнялась в элементе управления "Выбор даты и времени" (DTP). Это сообщение используется с кодом уведомления DTN_USERSTRING. (Юникод)

NMDATETIMEWMKEYDOWNA

Содержит сведения, используемые для описания и обработки кода уведомления DTN_WMKDOWN. (ANSI)

NMDATETIMEWMKEYDOWNW

Содержит сведения, используемые для описания и обработки кода уведомления DTN_WMKDOWN. (Юникод)

NMDAYSTATE

Содержит сведения, необходимые для обработки кода уведомления MCN_GETDAYSTATE. Все элементы этой структуры предназначены для входных данных, за исключением prgDayState, которые принимающее приложение должно задать при обработке MCN_GETDAYSTATE.

NMHDDISPINFOA

Содержит сведения, используемые для обработки кодов уведомлений HDN_GETDISPINFO. (ANSI)

NMHDDISPINFOW

Содержит сведения, используемые для обработки кодов уведомлений HDN_GETDISPINFO. (Юникод)

NMHDFILTERBTNCCLICK

Задает или получает атрибуты нажатия кнопки фильтра.

NMHEADERA

Содержит сведения о сообщениях уведомления элемента управления заголовком. Эта структура заменяет структуру HD_NOTIFY. (ANSI)

NMHEADERW

Содержит сведения о сообщениях уведомления элемента управления заголовком. Эта структура заменяет структуру HD_NOTIFY. (Юникод)

NMIPADDRESS

Содержит сведения для кода уведомления IPN_FIELDCHANGED.

NMITEMACTIVATE

Содержит сведения о коде уведомления LVN_ITEMACTIVATE.

NMKEY

Содержит сведения, используемые с ключевыми уведомлениями.

NMLINK

NMLINK содержит сведения об уведомлениях. Отправьте эту структуру с помощью сообщений NM_CLICK или NM_RETURN.

NMLISTVIEW

Содержит сведения об уведомлении представления списка. Эта структура аналогична структуре NM_LISTVIEW, но была переименована в соответствии со стандартными соглашениями об именовании.

NMLVCACHEHINT

Содержит сведения, используемые для обновления сведений об кэшированных элементах для использования с виртуальным представлением списка.

NMLVCUSTOMDRAW

Содержит сведения, относящиеся к коду уведомления NM_CUSTOMDRAW (представления списка), отправляемого элементом управления представлением списка.

NMLVDISPINFOA

Содержит сведения о коде уведомления LVN_GETDISPINFO или LVN_SETDISPINFO. Эта структура аналогична структуре LV_DISPINFO, но была переименована в соответствии со стандартными соглашениями об именовании. (ANSI)

NMLVDISPINFOW

Содержит сведения о коде уведомления LVN_GETDISPINFO или LVN_SETDISPINFO. Эта структура аналогична структуре LV_DISPINFO, но была переименована в соответствии со стандартными соглашениями об именовании. (Юникод)

NMLVEMPTYSKUP

Содержит сведения, используемые с кодом уведомления LVN_GETEMPTYSKUP.

NMLVFINDITEMA

Содержит сведения, необходимые владельцу для поиска элементов, запрашиваемых виртуальным элементом управления представлением списка. Эта структура используется с кодом уведомления LVN_ODFINDITEM. (ANSI)

NMLVFINDITEMW

Содержит сведения, необходимые владельцу для поиска элементов, запрашиваемых виртуальным элементом управления представлением списка. Эта структура используется с кодом уведомления LVN_ODFINDITEM. (Юникод)

NMLVGETINFOTIPA

Содержит и получает сведения об элементе представления списка, необходимые для отображения подсказки для элемента. Эта структура используется с кодом уведомления LVN_GETINFOTIP. (ANSI)

NMLVGETINFOTIPW

Содержит и получает сведения об элементе представления списка, необходимые для отображения подсказки для элемента. Эта структура используется с кодом уведомления LVN_GETINFOTIP. (Юникод)

NMLVKEYDOWN

Содержит сведения, используемые при обработке кода уведомления LVN_KEYDOWN. Эта структура аналогична структуре NMLVKEYDOWN, но была переименована в соответствии со стандартными соглашениями об именовании.

NMLVLINK

Содержит сведения о коде уведомления LVN_LINKCLICK.

NMLVODSTATECHANGE

Структура, содержащая сведения для использования при обработке кода уведомления LVN_ODSTATECHANGED.

NMLVSCROLL

Предоставляет сведения об операции прокрутки.

NMMOUSE

Содержит сведения, используемые с уведомлениями мыши.

NMOBJECTNOTIFY

Содержит сведения, используемые с кодами уведомлений TBN_GETOBJECT, TCN_GETOBJECT и PSN_GETOBJECT.

NMPGCALCSIZE

Содержит и получает сведения, используемые элементом управления пейджера для вычисления прокручиваемой области автономного окна. Он используется с уведомлением PGN_CALCSIZE.

NMPGHOTITEM

Содержит сведения, используемые с кодом уведомления PGN_HOTITEMCHANGE.

NMPGSCROLL

Содержит и получает сведения, которые элемент управления пейджера использует при прокрутке автономного окна. Он используется с уведомлением PGN_SCROLL.

NMRBAUTOSIZE

Содержит сведения, используемые для обработки кодов уведомлений RBN_AUTOSIZE.

NMREBAR

Содержит сведения, используемые для обработки различных уведомлений на панели.

NMREBARAUTOBREAK

Содержит сведения, используемые с кодом уведомления RBN_AUTOBREAK.

NMREBARCHEVRON

Содержит сведения, используемые для обработки кода уведомления RBN_CHEVRONPUSHED.

NMREBARCHILDSIZE

Содержит сведения, используемые для обработки кода уведомления RBN_CHILDSIZE.

NMREBARSPLITTER

Содержит сведения, используемые для обработки кода уведомления RBN_SPLITTERDRAG.

NMSEARCHWEB

Содержит сведения, используемые для обработки кода уведомления EN_SEARCHWEB .

NMSELCHANGE

Содержит сведения, необходимые для обработки кода уведомления MCN_SELCHANGE.

NMTBCUSTOMDRAW

Содержит сведения, относящиеся к коду уведомления NM_CUSTOMDRAW, отправляемым элементом управления панели инструментов.

NMTBDISPINFOA

Содержит и получает отображаемые сведения для элемента панели инструментов. Эта структура используется с кодом уведомления TBN_GETDISPINFO. (ANSI)

NMTBDISPINFO

Содержит и получает отображаемые сведения для элемента панели инструментов. Эта структура используется с кодом уведомления TBN_GETDISPINFO. (Юникод)

NMTBGETINFOTIPA

Содержит и получает сведения о подсказке для элемента панели инструментов. Эта структура используется с кодом уведомления TBN_GETINFOTIP. (ANSI)

NMTBGETINFOTIPW

Содержит и получает сведения о подсказке для элемента панели инструментов. Эта структура используется с кодом уведомления TBN_GETINFOTIP. (Юникод)

NMTBHOTITEM

Содержит сведения, используемые с кодом уведомления TBN_HOTITEMCHANGE.

NMTBRESTORE

Позволяет приложениям извлекать сведения, которые были помещены в NMTBSAVE при сохранении состояния панели инструментов. Эта структура передается приложениям, когда они получают код уведомления TBN_RESTORE.

NMTBSAVE

Эта структура передается приложениям при получении TBN_SAVE кода уведомления. Он содержит сведения о сохраняемой в данный момент кнопке. Приложения могут изменять значения членов для сохранения дополнительных сведений.

NMTCKEYDOWN

Содержит сведения о нажатии клавиши в элементе управления tab. Он используется с кодом уведомления TCN_KEYDOWN. Эта структура заменяет структуру TC_KEYDOWN.

NMTOOLBARA

Содержит сведения, используемые для обработки кодов уведомлений панели инструментов. Эта структура заменяет структуру TBNNOTIFY. (ANSI)

NMTOOLBARW

Содержит сведения, используемые для обработки кодов уведомлений панели инструментов. Эта структура заменяет структуру TBNNOTIFY. (Юникод)

NMTOOLTIPS CREATED

Содержит сведения, используемые с NM_TOOLTIPS CREATED кодами уведомлений.

NMTRBTHUMBPOSCHANGING

Содержит сведения об изменении панели отслеживания. Это сообщение отправляется вместе с уведомлением TRBN_THUMBPOSCHANGING.

NMTREEVIEWA

Содержит сведения о сообщении уведомления в виде дерева. Эта структура идентична структуре NM_TREEVIEW, но переименована в соответствии с текущими соглашениями об именовании. (ANSI)

NMTREEVIEWW

Содержит сведения о сообщении уведомления в виде дерева. Эта структура идентична структуре NM_TREEVIEW, но переименована в соответствии с текущими соглашениями об именовании. (Юникод)

NMTTCUSTOMDRAW

Содержит сведения, относящиеся к коду уведомления NM_CUSTOMDRAW, отправляемым элементом управления подсказки.

NMTTDISPINFOA

Содержит сведения, используемые для обработки кода уведомления TTN_GETDISPINFO. Эта структура заменяет структуру TOOLTIPTEXT. (ANSI)

NMTTDISPINFOW

Содержит сведения, используемые для обработки кода уведомления TTN_GETDISPINFO. Эта структура заменяет структуру TOOLTIPTEXT. (Юникод)

NMTVASYNCDRAW

Содержит объяснение причины сбоя рисования значка или элемента дерева наложения.

NMTVCUSTOMDRAW

Содержит сведения, относящиеся к NM_CUSTOMDRAW (древовидное представление) кода уведомления, отправляемого элементом управления в виде дерева.

NMTVDISPINFOA

Содержит и получает отображаемые сведения для элемента представления в виде дерева. Эта структура идентична структуре TV_DISPINFO, но она была переименована в соответствии с текущими соглашениями об именовании. (ANSI)

NMTVDISPINFOEXA

Содержит сведения, относящиеся к расширенным сведениям об уведомлениях TreeView.
(ANSI)

NMTVDISPINFOEXW

Содержит сведения, относящиеся к расширенным сведениям об уведомлениях TreeView.
(Юникод)

NMTVDISPINFOW

Содержит и получает отображаемые сведения для элемента представления в виде дерева.
Эта структура идентична структуре TV_DISPINFO, но она была переименована в соответствии
с текущими соглашениями об именовании. (Юникод)

NMTVGETINFOTIPA

Содержит и получает сведения об элементах в виде дерева, необходимые для отображения
подсказки для элемента. Эта структура используется с кодом уведомления TVN_GETINFOTIP.
(ANSI)

NMTVGETINFOTIPW

Содержит и получает сведения об элементах в виде дерева, необходимые для отображения
подсказки для элемента. Эта структура используется с кодом уведомления TVN_GETINFOTIP.
(Юникод)

NMTVITEMCHANGE

Содержит сведения об изменении элемента представления в виде дерева. Эта структура
отправляется вместе с уведомлениями TVN_ITEMCHANGED и TVN_ITEMCHANGING.

NMTVKEYDOWN

Содержит сведения о событии клавиатуры в элементе управления в виде дерева. Эта
структуре используется с кодом уведомления TVN_KEYDOWN. Структура идентична
структуре TV_KEYDOWN, но она была переименована в соответствии с текущими
соглашениями об именовании.

NMTVSTATEIMAGECHANGING

Содержит сведения о коде уведомления NM_TVSTATEIMAGECHANGING.

NMUPDOWN

Содержит сведения, относящиеся к уведомлениям элемента управления up-down. Он
идентичен и заменяет структуру NM_UPDOWN.

NMVIEWCHANGE

Хранит сведения, необходимые для обработки кода уведомления MCN_VIEWCHANGE.

PBRANGE

Содержит сведения о высоких и низких ограничениях элемента управления индикатора выполнения. Эта структура используется с сообщением PBM_GETRANGE.

RBHITTESTINFO

Содержит сведения, относящиеся к операции проверки попадания. Эта структура используется с сообщением RB_HITTEST.

REBARBANDINFOA

Содержит сведения, определяющие полосу в элементе управления rebar. (ANSI)

REBARBANDINFOW

Содержит сведения, определяющие полосу в элементе управления rebar. (Юникод)

REBARINFO

Содержит сведения, описывающие характеристики элемента управления rebar.

TASKDIALOG_BUTTON

Структура TASKDIALOG_BUTTON содержит сведения, используемые для отображения кнопки в диалоговом окне задачи. Эта структура используется в структуре TASKDIALOGCONFIG.

TASKDIALOGCONFIG

Структура TASKDIALOGCONFIG содержит сведения, используемые для отображения диалогового окна задачи. Функция TaskDialogIndirect использует эту структуру.

TBADD_BITMAP

Добавляет на панель инструментов растровое изображение, содержащее изображения кнопок.

TBBUTTON

Содержит сведения о кнопке на панели инструментов.

TBBUTTONINFOA

Содержит или получает сведения для определенной кнопки на панели инструментов. (ANSI)

TBBUTTONINFO

Содержит или получает сведения для определенной кнопки на панели инструментов.
(Юникод)

TBINSETMARK

Содержит сведения о метке вставки в элементе управления панели инструментов.

МЕТРИКИ ТБ

Определяет метрики панели инструментов, используемые для сжатия или развертывания элементов панели инструментов.

TBREPLACEBITMAP

Используется с сообщением TB_REPLACEBITMAP для замены одного растрового изображения панели инструментов другим.

TBSAVEPARAMSA

Указывает расположение в реестре, в котором TB_SAVERESTORE сообщение хранит и получает сведения о состоянии панели инструментов. (ANSI)

TBSAVEPARAMSW

Указывает расположение в реестре, в котором TB_SAVERESTORE сообщение хранит и получает сведения о состоянии панели инструментов. (Юникод)

TCHITTESTINFO

Содержит сведения о проверке нажатия. Эта структура заменяет структуру TC_HITTESTINFO.

TCITEMA

Задает или получает атрибуты элемента вкладки. Он используется с TCM_INSERTITEM, TCM_GETITEM и TCM_SETITEM сообщениями. Эта структура заменяет структуру TC_ITEM.
(ANSI)

TCITEMHEADERA

Задает или получает атрибуты вкладки. Он используется с TCM_INSERTITEM, TCM_GETITEM и TCM_SETITEM сообщениями. Эта структура заменяет структуру TC_ITEMHEADER.
(ANSI)

TCITEMHEADERW

Задает или получает атрибуты вкладки. Он используется с TCM_INSERTITEM, TCM_GETITEM и TCM_SETITEM сообщениями. Эта структура заменяет структуру TC_ITEMHEADER.
(Юникод)

[TCITEMW](#)

Задает или получает атрибуты элемента вкладки. Он используется с TCM_INSERTITEM, TCM_GETITEM и TCM_SETITEM сообщениями. Эта структура заменяет структуру TC_ITEM. (Юникод)

[TTGETTITLE](#)

Предоставляет сведения о заголовке элемента управления подсказкой.

[TTHITTESTINFOA](#)

Содержит сведения, которые элемент управления подсказкой использует для определения того, находится ли точка в ограничивающем прямоугольнике указанного инструмента. Если точка находится в прямоугольнике, структура получает сведения об инструменте. (ANSI)

[TTHITTESTINFOW](#)

Содержит сведения, которые элемент управления подсказкой использует для определения того, находится ли точка в ограничивающем прямоугольнике указанного инструмента. Если точка находится в прямоугольнике, структура получает сведения об инструменте. (Юникод)

[TTTOOLINFOA](#)

Структура TOOLINFO содержит сведения об инструменте в элементе управления подсказкой. (ANSI)

[TTTOOLINFOW](#)

Структура TOOLINFO содержит сведения об инструменте в элементе управления подсказкой. (Юникод)

[TVGETITEMPARTRECTINFO](#)

Содержит сведения для идентификации "зоны попадания" для указанной части элемента дерева. Структура используется с сообщением TVM_GETITEMPARTRECT и макросом TreeView_GetItemPartRect.

[TVHITTESTINFO](#)

Содержит сведения, используемые для определения расположения точки относительно элемента управления в виде дерева.

[TVINSERTSTRUCTA](#)

Содержит сведения, используемые для добавления нового элемента в элемент управления в виде дерева. Эта структура используется с сообщением TVM_INSERTITEM. Структура идентична структуре TV_INSERTSTRUCT, но переименована в соответствии с текущими соглашениями об именовании. (ANSI)

[TVINSERTSTRUCTW](#)

Содержит сведения, используемые для добавления нового элемента в элемент управления в виде дерева. Эта структура используется с сообщением TVM_INSERTITEM. Структура идентична структуре TV_INSERTSTRUCT, но переименована в соответствии с текущими соглашениями об именовании. (Юникод)

[TVITEMA](#)

Задает или получает атрибуты элемента представления в виде дерева. Эта структура идентична структуре TV_ITEM, но она была переименована в соответствии с текущими соглашениями об именовании. Новые приложения должны использовать эту структуру. (ANSI)

[TVITEMEXA](#)

Задает или получает атрибуты элемента представления в виде дерева. Эта структура является улучшением структуры TVITEM. Новые приложения должны использовать эту структуру, если это уместно. (ANSI)

[TVITEMEXW](#)

Задает или получает атрибуты элемента представления в виде дерева. Эта структура является улучшением структуры TVITEM. Новые приложения должны использовать эту структуру, если это уместно. (Юникод)

[TVITEMW](#)

Задает или получает атрибуты элемента представления в виде дерева. Эта структура идентична структуре TV_ITEM, но она была переименована в соответствии с текущими соглашениями об именовании. Новые приложения должны использовать эту структуру. (Юникод)

[TVSORTCB](#)

Содержит сведения, используемые для сортировки дочерних элементов в элементе управления в виде дерева. Эта структура используется с сообщением TVM_SORTCHILDRENCB. Эта структура идентична структуре TV_SORTCB, но она была переименована в соответствии с текущими соглашениями об именовании.

UDACCEL

Содержит сведения об ускорении для элемента управления "Вверх-вниз".

Перечисления

EC_ENDOFLINE

Указывает конец символа строки, используемого элементом управления редактированием.

EC_SEARCHWEB_ENTRYPOINT

Определяет константы, указывающие точку входа поиска в Интернете.

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция _TrackMouseEvent (commctrl.h)

Статья 22.08.2023

Публикует сообщения, когда указатель мыши покидает окно или наносит указатель мыши на окно в течение указанного периода времени. Эта функция вызывает [TrackMouseEvent](#), если она существует, в противном случае эмулирует ее.

Синтаксис

C++

```
BOOL _TrackMouseEvent(  
    [in, out] LPTRACKMOUSEEVENT lpEventTrack  
);
```

Параметры

[in, out] lpEventTrack

Тип: [LPTRACKMOUSEEVENT](#)

Указатель на структуру [TRACKMOUSEEVENT](#), содержащую сведения об отслеживании.

Возвращаемое значение

Тип: [BOOL](#)

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция завершается сбоем, возвращаемое значение равно нулю.

Требования

Минимальная версия
клиента

Windows 2000 Professional [только классические
приложения]

Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	commctrl.h
Библиотека	Comctl32.lib
DLL	Comctl32.dll

См. также раздел

[Основные понятия](#)

[Ввод с помощью мыши](#)

[Другие ресурсы](#)

[Справочные материалы](#)

[SystemParametersInfo](#)

[TRACKMOUSEEVENT](#)

[TrackMouseEvent](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Заголовок winuser.h

Статья 08.03.2023

Этот заголовок используется несколькими технологиями. Дополнительные сведения см. в разделе:

- [Обмен данными](#)
- [Диспетчер окон рабочего стола \(DWM\)](#)
- [Заметки для разработчиков](#)
- [Диалоговые окна](#)
- [Справочник по отображаемым устройствам](#)
- [Приложения с высоким DPI](#)
- [Конфигурация обратной связи ввода](#)
- [Идентификация источника входных данных](#)
- [Интернационализация для приложений Windows](#)
- [Ввод с помощью клавиатуры и мыши](#)
- [Меню и другие ресурсы](#)
- [Поставщик параметров мобильного Управление устройствами](#)
- [Стек ввода указателя устройства](#)
- [Сообщения и уведомления ввода указателя](#)
- [Службы удаленных рабочих столов](#)
- [Безопасность и идентификация](#)
- [Системные службы](#)
- [Оболочка Windows](#)
- [Тестирование нажатия касания](#)
- [Внедрение сенсорного ввода](#)
- [Сенсорный ввод](#)
- [Оконные станции и настольные компьютеры](#)
- [Специальные возможности Windows](#)
- [Windows и сообщения](#)
- [Элементы управления Windows](#)
- [Windows GDI](#)

Winuser.h содержит следующие программные интерфейсы:

ФУНКЦИИ



[ActivateKeyboardLayout](#)

Задает идентификатор входного языкового стандарта (прежнее название — дескриптор раскладки клавиатуры) для вызывающего потока или текущего процесса. Идентификатор входного языкового стандарта указывает языковой стандарт, а также физический макет клавиатуры.

[AddClipboardFormatListener](#)

Помещает заданное окно в список прослушивателя в формате буфера обмена, поддерживаемый системой.

[AdjustWindowRect](#)

Вычисляет требуемый размер прямоугольника окна на основе требуемого размера клиентского прямоугольника. Затем прямоугольник окна можно передать в функцию CreateWindow, чтобы создать окно с клиентской областью требуемого размера.

[AdjustWindowRectEx](#)

Вычисляет требуемый размер прямоугольника окна на основе требуемого размера клиентского прямоугольника. Затем прямоугольник окна можно передать в функцию CreateWindowEx, чтобы создать окно с клиентской областью требуемого размера.

[AdjustWindowRectExForDpi](#)

Вычисляет требуемый размер прямоугольника окна на основе требуемого размера клиентского прямоугольника и предоставленного DPI.

[AllowSetForegroundWindow](#)

Позволяет указанному процессу задать окно переднего плана с помощью функции SetForegroundWindow. Вызывающий процесс уже должен иметь возможность задать окно переднего плана. Дополнительные сведения см. в подразделе «Примечания» далее в этом разделе.

[AnimateWindow](#)

Позволяет создавать специальные эффекты при отображении или скрытии окон. Существует четыре типа анимации: _roll, скольжения, свертывания или развертывания и альфа-смешивания.

[AnyPopup](#)

Указывает, существует ли на экране собственное, видимое, всплывающее окно верхнего уровня или перекрывающееся окно. Функция выполняет поиск по всему экрану, а не только в клиентской области вызывающего приложения.

[AppendMenuA](#)

Добавляет новый элемент в конец указанной строки меню, раскрывающегося меню, подменю или контекстного меню. Эту функцию можно использовать для указания содержимого, внешнего вида и поведения элемента меню. (ANSI)

[AppendMenuW](#)

Добавляет новый элемент в конец указанной строки меню, раскрывающегося меню, подменю или контекстного меню. Эту функцию можно использовать для указания содержимого, внешнего вида и поведения элемента меню. (Юникод)

[AreDpiAwarenessContextsEqual](#)

Определяет, совпадают ли два DPI_AWARENESS_CONTEXT значения.

[ArrangeIconicWindows](#)

Упорядочивает все свернутые (знаковые) дочерние окна указанного родительского окна.

[AttachThreadInput](#)

Присоединяет или отсоединяет входной механизм обработки одного потока к механизму обработки другого потока.

[BeginDeferWindowPos](#)

Выделяет память для структуры с несколькими окнами и возвращает дескриптор в структуру.

[BeginPaint](#)

Функция BeginPaint готовит указанное окно для рисования и заполняет структуру PAINTSTRUCT сведениями о покраски.

[BlockInput](#)

Блокирует доступ к приложениям с помощью клавиатуры и мыши.

[BringWindowToFront](#)

Переносит указанное окно в начало порядка Z. Если окно является окном верхнего уровня, оно активируется. Если окно является дочерним, активируется родительское окно верхнего уровня, связанное с дочерним окном.

[BroadcastSystemMessage](#)

Функция BroadcastSystemMessage отправляет сообщение указанным получателям. (BroadcastSystemMessage)

[BroadcastSystemMessageA](#)

Отправляет сообщение указанным получателям. (`BroadcastSystemMessageA`)

[BroadcastSystemMessageExA](#)

Отправляет сообщение указанным получателям. (`BroadcastSystemMessageExA`)

[BroadcastSystemMessageExW](#)

Отправляет сообщение указанным получателям. (`BroadcastSystemMessageExW`)

[BroadcastSystemMessageW](#)

Функция `BroadcastSystemMessageW` (Юникод) отправляет сообщение указанным получателям. (`BroadcastSystemMessageW`)

[CalculatePopupWindowPosition](#)

Вычисляет соответствующую позицию всплывающего окна, используя указанную точку привязки, размер всплывающего окна, флаги и необязательный прямоугольник исключения.

[CallIMsgFilterA](#)

Передает указанные сообщения и код перехватчика в процедуры перехватчика, связанные с `WH_SYSMSGFILTER` и `WH_MSGFILTER` перехватчиками. (ANSI)

[CallIMsgFilterW](#)

Передает указанное сообщение и код перехватчика в процедуры перехватчика, связанные с `WH_SYSMSGFILTER` и `WH_MSGFILTER` перехватчиками. (Юникод)

[CallNextHookEx](#)

Передает сведения о перехватчике в следующую процедуру перехватчика в текущей цепочке перехватчиков. Процедура перехватчика может вызывать эту функцию до или после обработки сведений о перехватчике.

[CallWindowProcA](#)

Передает сведения о сообщении в указанную процедуру окна. (ANSI)

[CallWindowProcW](#)

Передает сведения о сообщении в указанную процедуру окна. (Юникод)

[CascadeWindows](#)

Каскадирует указанные дочерние окна указанного родительского окна.

[Цепочка changeClipboardChain](#)

Удаляет указанное окно из цепочки средств просмотра буфера обмена.

[ChangeDisplaySettingsA](#)

Функция ChangeDisplaySettings изменяет параметры устройства отображения по умолчанию на указанный графический режим. (ANSI)

[ChangeDisplaySettingsExA](#)

Функция ChangeDisplaySettingsEx изменяет параметры указанного устройства отображения на указанный режим графики. (ANSI)

[ChangeDisplaySettingsExW](#)

Функция ChangeDisplaySettingsEx изменяет параметры указанного устройства отображения на указанный режим графики. (Юникод)

[ChangeDisplaySettingsW](#)

Функция ChangeDisplaySettings изменяет параметры устройства отображения по умолчанию на указанный графический режим. (Юникод)

[ChangeWindowMessageFilter](#)

Добавляет или удаляет сообщение из фильтра изоляции привилегий пользовательского интерфейса (UIPI).

[ChangeWindowMessageFilterEx](#)

Изменяет фильтр сообщений пользовательского интерфейса Privilege Isolation (UIPI) для указанного окна.

[CharLowerA](#)

Преобразует строку символов или один символ в нижний регистр. Если операнд является строкой символов, функция преобразует символы на месте. (ANSI)

[CharLowerBuffA](#)

Преобразует символы верхнего регистра в буфере в символы нижнего регистра. Функция преобразует символы на месте. (ANSI)

[CharLowerBuffW](#)

Преобразует символы верхнего регистра в буфере в символы нижнего регистра. Функция преобразует символы на месте. (Юникод)

[CharLowerW](#)

Преобразует строку символов или один символ в нижний регистр. Если операнд является строкой символов, функция преобразует символы на месте. (Юникод)

[CharNextA](#)

Извлекает указатель на следующий символ в строке. Эта функция может обрабатывать строки, состоящие из однобайтовых или многобайтовых символов. (ANSI)

[CharNextExA](#)

Извлекает указатель на следующий символ в строке. Эта функция может обрабатывать строки, состоящие из однобайтовых или многобайтовых символов.

[CharNextW](#)

Извлекает указатель на следующий символ в строке. Эта функция может обрабатывать строки, состоящие из однобайтовых или многобайтовых символов. (Юникод)

[CharPrevA](#)

Извлекает указатель на предыдущий символ в строке. Эта функция может обрабатывать строки, состоящие из однобайтовых или многобайтовых символов. (ANSI)

[CharPrevExA](#)

Извлекает указатель на предыдущий символ в строке. Эта функция может обрабатывать строки, состоящие из однобайтовых или многобайтовых символов.

[CharPrevW](#)

Извлекает указатель на предыдущий символ в строке. Эта функция может обрабатывать строки, состоящие из однобайтовых или многобайтовых символов. (Юникод)

[CharToOemA](#)

Преобразует строку в определяемую изготовителем оборудования кодировку.
Предупреждение Не используйте. (ANSI)

[CharToOemBuffA](#)

Преобразует указанное число символов в строке в набор символов, определяемый OEM.
(ANSI)

[CharToOemBuffW](#)

Преобразует указанное число символов в строке в набор символов, определяемый OEM.
(Юникод)

[CharToOemW](#)

Преобразует строку в определяемую изготовителем оборудования кодировку.
Предупреждение Не используйте. (Юникод)

[CharUpperA](#)

Преобразует строку символов или один символ в верхний регистр. Если операнд является строкой символов, функция преобразует символы на месте. (ANSI)

[CharUpperBuffA](#)

Преобразует символы нижнего регистра в буфере в символы верхнего регистра. Функция преобразует символы на месте. (ANSI)

[CharUpperBuffW](#)

Преобразует символы нижнего регистра в буфере в символы верхнего регистра. Функция преобразует символы на месте. (Юникод)

[CharUpperW](#)

Преобразует строку символов или один символ в верхний регистр. Если операнд является строкой символов, функция преобразует символы на месте. (Юникод)

[CheckDlgButton](#)

Изменяет состояние проверка элемента управления "Кнопка".

[CheckMenuItem](#)

Устанавливает состояние атрибута проверка метки указанного пункта меню в выбранном или чистом состоянии.

[CheckMenuRadioItem](#)

Проверяет указанный пункт меню и делает его элементом-переключателем. В то же время функция очищает все остальные пункты меню в связанной группе и очищает флаг типа переключателя для этих элементов.

[CheckRadioButton](#)

Добавляет метку проверка в (проверяет) указанный переключатель в группе и удаляет метку проверка из всех остальных переключателей в группе.

[ChildWindowFromPoint](#)

Определяет, какое из дочерних окон, принадлежащих родительскому окну, содержит указанную точку , если таковое имеется. Поиск ограничен непосредственными дочерними окнами. Внуки, и более глубокие окна потомков не ищутся.

[ChildWindowFromPointEx](#)

Определяет, какое из дочерних окон, принадлежащих указанному родительскому окну, содержит указанную точку (если таковое есть).

[ClientToScreen](#)

Функция ClientToScreen преобразует координаты клиентской области указанной точки в экранные координаты.

[ClipCursor](#)

Ограничивает курсор прямоугольной областью на экране.

[CloseClipboard](#)

Закрывает буфер обмена.

[CloseDesktop](#)

Закрывает открытый дескриптор для объекта рабочего стола.

[CloseGestureInfoHandle](#)

Закрывает ресурсы, связанные с дескриптором сведений о жестах.

[CloseTouchInputHandle](#)

Закрывает дескриптор сенсорного ввода, освобождает связанную с ним память процесса и делает дескриптор недействительным.

[CloseWindow](#)

Сворачивать (но не уничтожать) указанное окно.

[CloseWindowStation](#)

Закрывает открытый дескриптор оконной станции.

[CopyAcceleratorTableA](#)

Копирует указанную таблицу ускорителей. Эта функция используется для получения данных таблицы ускорителя, соответствующей дескриптору таблицы ускорителя, или для определения размера данных таблицы ускорителя. (ANSI)

[CopyAcceleratorTableW](#)

Копирует указанную таблицу ускорителей. Эта функция используется для получения данных таблицы ускорителя, соответствующей дескриптору таблицы ускорителя, или для определения размера данных таблицы ускорителя. (Юникод)

[CopyCursor](#)

Копирует указанный курсор.

[CopyIcon](#)

Копирует указанный значок из другого модуля в текущий модуль.

[CopyImage](#)

Создает новое изображение (значок, курсор или растровое изображение) и копирует атрибуты указанного изображения в новое изображение. При необходимости функция растягивает биты в соответствии с требуемым размером нового изображения.

[CopyRect](#)

Функция CopyRect копирует координаты одного прямоугольника в другой.

[CountClipboardFormats](#)

Извлекает количество различных форматов данных, которые в настоящее время находятся в буфере обмена.

[CreateAcceleratorTableA](#)

Создает таблицу ускорителей. (ANSI)

[CreateAcceleratorTableW](#)

Создает таблицу ускорителей. (Юникод)

[CreateCaret](#)

Создает новую фигуру для системного курсора и назначает владение курсором заданному окну. Фигура курсора может быть линией, блоком или растровым рисунком.

[CreateCursor](#)

Создает курсор, имеющий указанный размер, битовые шаблоны и горячую точку.

[CreateDesktopA](#)

Создает новый рабочий стол, связывает его с текущей станцией окна вызывающего процесса и назначает его вызывающему потоку. (ANSI)

[CreateDesktopExA](#)

Создает рабочий стол с указанной кучей, связывает его с текущей станцией окна вызывающего процесса и назначает его вызывающему потоку. (ANSI)

[CreateDesktopExW](#)

Создает рабочий стол с указанной кучей, связывает его с текущей станцией окна вызывающего процесса и назначает его вызывающему потоку. (Юникод)

[CreateDesktopW](#)

Создает новый рабочий стол, связывает его с текущей станцией окна вызывающего процесса и назначает его вызывающему потоку. (Юникод)

[CreateDialogA](#)

Создает немодерное диалоговое окно из ресурса шаблона диалогового окна. Макрос CreateDialog использует функцию CreateDialogParam. (ANSI)

[CreateDialogIndirectA](#)

Создает немодерное диалоговое окно на основе шаблона диалогового окна в памяти. Макрос CreateDialogIndirect использует функцию CreateDialogIndirectParam. (ANSI)

[CreateDialogIndirectParamA](#)

Создает немодерное диалоговое окно на основе шаблона диалогового окна в памяти. (ANSI)

[CreateDialogIndirectParamW](#)

Создает немодерное диалоговое окно на основе шаблона диалогового окна в памяти. (Юникод)

[CreateDialogIndirectW](#)

Создает немодерное диалоговое окно на основе шаблона диалогового окна в памяти. Макрос CreateDialogIndirect использует функцию CreateDialogIndirectParam. (Юникод)

[CreateDialogParamA](#)

Создает немодерное диалоговое окно из ресурса шаблона диалогового окна. (ANSI)

[CreateDialogParamW](#)

Создает немодерное диалоговое окно из ресурса шаблона диалогового окна. (Юникод)

[CreateDialogW](#)

Создает немодерное диалоговое окно из ресурса шаблона диалогового окна. Макрос CreateDialog использует функцию CreateDialogParam. (Юникод)

[CreateIcon](#)

Создает значок с указанным размером, цветами и узорами битов.

[CreateIconFromResource](#)

Создает значок или курсор из битов ресурсов, описывающих значок.
(CreateIconFromResource)

[CreateIconFromResourceEx](#)

Создает значок или курсор из битов ресурсов, описывающих значок.
(CreateIconFromResourceEx)

[CreateIconIndirect](#)

Создает значок или курсор из структуры ICONINFO.

[CreateMDIWindowA](#)

Создает дочернее окно многодокументного интерфейса (MDI). (ANSI)

[CreateMDIWindowW](#)

Создает дочернее окно многодокументного интерфейса (MDI). (Юникод)

[CreateMenu](#)

Создает меню. Изначально меню пустое, но его можно заполнить элементами меню с помощью функций InsertMenuItem, AppendMenu и InsertMenu.

[CreatePopupMenu](#)

Создает раскрывающееся меню, подменю или контекстное меню.

[CreateSyntheticPointerDevice](#)

Настраивает устройство внедрения указателя для вызывающего приложения и инициализирует максимальное количество одновременных указателей, которые приложение может внедрять.

[CreateWindowA](#)

Создает перекрывающееся, всплывающее или дочернее окно. (ANSI)

[CreateWindowExA](#)

Создает перекрывающееся, всплывающее или дочернее окно с расширенным стилем окна; В противном случае эта функция идентична функции CreateWindow. (ANSI)

[CreateWindowExW](#)

Создает перекрывающееся, всплывающее или дочернее окно с расширенным стилем окна; В противном случае эта функция идентична функции CreateWindow. (Юникод)

[CreateWindowStationA](#)

Создает объект оконной станции, связывает его с вызывающим процессом и назначает его текущему сеансу. (ANSI)

[CreateWindowStationW](#)

Создает объект оконной станции, связывает его с вызывающим процессом и назначает его текущему сеансу. (Юникод)

[CreateWindowW](#)

Создает перекрывающееся, всплывающее или дочернее окно. (Юникод)

[DefDlgProcA](#)

Вызывает процедуру диалогового окна по умолчанию, чтобы обеспечить обработку по умолчанию для всех оконных сообщений, которые не обрабатываются диалоговым окном с закрытым классом окна. (ANSI)

[DefDlgProcW](#)

Вызывает процедуру диалогового окна по умолчанию, чтобы обеспечить обработку по умолчанию для всех оконных сообщений, которые не обрабатываются диалоговым окном с закрытым классом окна. (Юникод)

[DeferWindowPos](#)

Обновления указанную структуру позиции нескольких окон для указанного окна.

[DefFrameProcA](#)

Обеспечивает обработку по умолчанию для любых оконных сообщений, которые не обрабатываются процедурой окна фрейма многодокументного интерфейса (MDI). (ANSI)

[DefFrameProcW](#)

Обеспечивает обработку по умолчанию для любых оконных сообщений, которые не обрабатываются процедурой окна фрейма многодокументного интерфейса (MDI). (Юникод)

[DefMDIChildProcA](#)

Обеспечивает обработку по умолчанию для любого сообщения окна, которое не обрабатывается процедурой окна дочернего окна с несколькими документами (MDI). (ANSI)

[DefMDIChildProcW](#)

Обеспечивает обработку по умолчанию для любого сообщения окна, которое не обрабатывается процедурой окна дочернего окна с несколькими документами (MDI). (Юникод)

[DefRawInputProc](#)

Проверяет правильность размера структуры RAWINPUTHEADER.

[DefWindowProcA](#)

Вызывает процедуру окна по умолчанию, чтобы обеспечить обработку по умолчанию для всех оконных сообщений, которые не обрабатываются приложением. (ANSI)

[DefWindowProcW](#)

Вызывает процедуру окна по умолчанию, чтобы обеспечить обработку по умолчанию для всех оконных сообщений, которые не обрабатываются приложением. (Юникод)

[DeleteMenu](#)

Удаляет элемент из указанного меню. Если пункт меню открывает меню или вложенное меню, эта функция удаляет дескриптор в меню или подменю и освобождает память, используемую меню или подменю.

[Отмена регистрацииShellHookWindow](#)

Отменяет регистрацию указанного окна оболочки, которое зарегистрировано для получения сообщений обработчика оболочки.

[DestroyAcceleratorTable](#)

Уничтожает таблицу ускорителей.

[DestroyCaret](#)

Уничтожает текущую форму курсора, освобождает курсор из окна и удаляет курсор с экрана.

[DestroyCursor](#)

Уничтожает курсор и освобождает память, занятую курсором. Не используйте эту функцию для уничтожения общего курсора.

[DestroyIcon](#)

Уничтожает значок и освобождает память, занятую значком.

[DestroyMenu](#)

Уничтожает указанное меню и освобождает память, занимаемую меню.

[DestroySyntheticPointerDevice](#)

Уничтожает указанное устройство внедрения указателя.

[DestroyWindow](#)

Уничтожает указанное окно.

[DialogBoxA](#)

Создает модальное диалоговое окно из ресурса шаблона диалогового окна. DialogBox не возвращает управление, пока указанная функция обратного вызова не завершит модальное диалоговое окно путем вызова функции EndDialog. (ANSI)

[DialogBoxIndirectA](#)

Создает модальное диалоговое окно на основе шаблона диалогового окна в памяти. DialogBoxIndirect не возвращает управление, пока указанная функция обратного вызова не завершит модальное диалоговое окно, вызвав функцию EndDialog. (ANSI)

[DialogBoxIndirectParamA](#)

Создает модальное диалоговое окно на основе шаблона диалогового окна в памяти. (ANSI)

[DialogBoxIndirectParamW](#)

Создает модальное диалоговое окно на основе шаблона диалогового окна в памяти. (Юникод)

[DialogBoxIndirectW](#)

Создает модальное диалоговое окно на основе шаблона диалогового окна в памяти. DialogBoxIndirect не возвращает управление, пока указанная функция обратного вызова не завершит модальное диалоговое окно, вызвав функцию EndDialog. (Юникод)

[DialogBoxParamA](#)

Создает модальное диалоговое окно из ресурса шаблона диалогового окна. (ANSI)

[DialogBoxParamW](#)

Создает модальное диалоговое окно из ресурса шаблона диалогового окна. (Юникод)

[DialogBoxW](#)

Создает модальное диалоговое окно из ресурса шаблона диалогового окна. DialogBox не возвращает управление, пока указанная функция обратного вызова не завершит модальное диалоговое окно путем вызова функции EndDialog. (Юникод)

[DisableProcessWindowsGhosting](#)

Отключает функцию создания фантомного окна для процесса графического пользовательского интерфейса вызова. Фантомное создание окон — это функция Диспетчера Windows, которая позволяет пользователю свернуть, переместить или закрыть main окно приложения, которое не отвечает.

[DispatchMessage](#)

Функция DispatchMessage отправляет сообщение в процедуру окна. Обычно он используется для отправки сообщения, полученного функцией GetMessage.

[DispatchMessageA](#)

Отправляет сообщение в процедуру окна. Обычно он используется для отправки сообщения, полученного функцией GetMessage. (DispatchMessageA)

[DispatchMessageW](#)

Функция DispatchMessageW (Юникод) отправляет сообщение в процедуру окна. Обычно он используется для отправки сообщения, полученного функцией GetMessage.

[DisplayConfigGetDeviceInfo](#)

Функция DisplayConfigGetDeviceInfo извлекает сведения о конфигурации отображения устройства.

[DisplayConfigSetDeviceInfo](#)

Функция `DisplayConfigSetDeviceInfo` задает свойства целевого объекта.

[DlgDirListA](#)

Заменяет содержимое списка именами подкаталогов и файлов в указанном каталоге. Список имен можно отфильтровать, указав набор атрибутов файла. При необходимости в список могут входить сопоставленные диски. (ANSI)

[DlgDirListComboBoxA](#)

Заменяет содержимое поля со списком именами подкаталогов и файлов в указанном каталоге. Список имен можно отфильтровать, указав набор атрибутов файла. Список имен может включать сопоставленные буквы дисков. (ANSI)

[DlgDirListComboBoxW](#)

Заменяет содержимое поля со списком именами подкаталогов и файлов в указанном каталоге. Список имен можно отфильтровать, указав набор атрибутов файла. Список имен может включать сопоставленные буквы дисков. (Юникод)

[DlgDirListW](#)

Заменяет содержимое списка именами подкаталогов и файлов в указанном каталоге. Список имен можно отфильтровать, указав набор атрибутов файла. При необходимости в список могут входить сопоставленные диски. (Юникод)

[DlgDirSelectComboBoxExA](#)

Извлекает текущий фрагмент из поля со списком, заполненного с помощью функции `DlgDirListComboBox`. Выбор интерпретируется как буква диска, файл или имя каталога. (ANSI)

[DlgDirSelectComboBoxExW](#)

Извлекает текущий фрагмент из поля со списком, заполненного с помощью функции `DlgDirListComboBox`. Выбор интерпретируется как буква диска, файл или имя каталога. (Юникод)

[DlgDirSelectExA](#)

Извлекает текущий выделенный фрагмент из списка с одним выбором. Предполагается, что поле списка было заполнено функцией `DlgDirList` и выбрано буква диска, имя файла или имя каталога. (ANSI)

[DlgDirSelectExW](#)

Извлекает текущий выделенный фрагмент из списка с одним выбором. Предполагается, что поле списка было заполнено функцией DlgDirList и выбрано буква диска, имя файла или имя каталога. (Юникод)

[DragDetect](#)

Захватывает мышь и отслеживает ее движение, пока пользователь не отпустит левую кнопку мыши, не нажмет клавишу ESC или не переместит мышь за пределы прямоугольника перетаскивания, в котором находится указанная точка.

[DrawAnimatedRects](#)

Анимирует подпись окна, указывая на открытие значка, свертывание или увеличение окна.

[DrawCaption](#)

Функция DrawCaption рисует окно подпись.

[DrawEdge](#)

Функция DrawEdge рисует один или несколько краев прямоугольника.

[DrawFocusRect](#)

Функция DrawFocusRect рисует прямоугольник в стиле, который указывает, что прямоугольник имеет фокус.

[DrawFrameControl](#)

Функция DrawFrameControl рисует элемент управления кадром указанного типа и стиля.

[DrawIcon](#)

Рисует значок или курсор в указанный контекст устройства.

[DrawIconEx](#)

Рисует значок или курсор в указанный контекст устройства, выполняя указанные растровые операции и растягивая или сжимая значок или курсор, как указано.

[DrawMenuBar](#)

Перерисовывает строку меню указанного окна. Если строка меню изменяется после создания окна системой, эту функцию необходимо вызвать для рисования измененной строки меню.

[DrawStateA](#)

Функция DrawState отображает изображение и применяет визуальный эффект, чтобы указать состояние, например отключенное или стандартное. (ANSI)

[DrawStateW](#)

Функция DrawState отображает изображение и применяет визуальный эффект, чтобы указать состояние, например отключенное или стандартное. (Юникод)

[Drawtext](#)

Функция DrawText рисует форматированный текст в указанном прямоугольнике. (Функция DrawText)

[DrawTextA](#)

Функция DrawText рисует форматированный текст в указанном прямоугольнике. Он форматирует текст в соответствии с указанным методом (разворачивание вкладок, обоснование символов, разрыв линий и т. д.). (DrawTextA)

[DrawTextExA](#)

Функция DrawTextEx рисует отформатированный текст в указанном прямоугольнике. (ANSI)

[DrawTextExW](#)

Функция DrawTextEx рисует отформатированный текст в указанном прямоугольнике. (Юникод)

[DrawTextW](#)

Функция DrawTextW (Юникод) рисует отформатированный текст в указанном прямоугольнике. (функция DrawTextW)

[EmptyClipboard](#)

Очищает буфер обмена и освобождает дескрипторы данных в буфере обмена. Затем функция назначает право владения буфером обмена окну, в которое в настоящее время открыт буфер обмена.

[EnableMenuItem](#)

Включает, отключает или серым цветом указанный пункт меню.

[EnableMouseInPointer](#)

Позволяет мыши выступать в качестве устройства ввода указателя и отправлять WM_POINTER сообщения.

[EnableNonClientDpiScaling](#)

На дисплеях с высоким разрешением включает автоматическое масштабирование отображения частей неклиентской области указанного окна верхнего уровня. Должен вызываться во время инициализации этого окна.

[EnableScrollBar](#)

Функция EnableScrollBar включает или отключает одну или обе стрелки полосы прокрутки.

[EnableWindow](#)

Включает или отключает ввод с помощью мыши и клавиатуры для указанного окна или элемента управления. Если входные данные отключены, окно не получает такие входные данные, как щелчки мышью и нажатия клавиш. Если вход включен, окно получает все входные данные.

[EndDeferWindowPos](#)

Одновременно обновляет положение и размер одного или нескольких окон за один цикл обновления экрана.

[EndDialog](#)

Уничтожает модальное диалоговое окно, в результате чего система завершает обработку диалогового окна.

[EndMenu](#)

Завершает активное меню вызывающего потока.

[EndPaint](#)

Функция EndPaint помечает конец рисования в указанном окне. Эта функция требуется для каждого вызова функции BeginPaint, но только после завершения рисования.

[EndTask](#)

Принудительно закрывает указанное окно.

[EnumChildWindows](#)

Перечисляет дочерние окна, принадлежащие указанному родительскому окну, передавая дескриптор каждому дочернему окну, в свою очередь, в определяемую приложением функцию обратного вызова.

[EnumClipboardFormats](#)

Перечисляет форматы данных, доступные в настоящее время в буфере обмена.

[EnumDesktopsA](#)

Перечисляет все рабочие столы, связанные с указанной оконной станцией вызывающего процесса. Функция, в свою очередь, передает имя каждого рабочего стола в определяемую приложением функцию обратного вызова. (ANSI)

[EnumDesktopsW](#)

Перечисляет все рабочие столы, связанные с указанной оконной станцией вызывающего процесса. Функция, в свою очередь, передает имя каждого рабочего стола в определяемую приложением функцию обратного вызова. (Юникод)

[EnumDesktopWindows](#)

Перечисляет все окна верхнего уровня, связанные с указанным рабочим столом. Он передает дескриптор каждому окну, в свою очередь, в определяемую приложением функцию обратного вызова.

[EnumDisplayDevicesA](#)

Функция EnumDisplayDevices позволяет получать сведения об устройствах отображения в текущем сеансе. (ANSI)

[EnumDisplayDevicesW](#)

Функция EnumDisplayDevices позволяет получать сведения об устройствах отображения в текущем сеансе. (Юникод)

[EnumDisplayMonitors](#)

Функция EnumDisplayMonitors перечисляет мониторы отображения (включая невидимые псевдомониторы, связанные с драйверами зеркального отображения), которые пересекают область, сформированную пересечением указанного прямоугольника обрезки и видимой области контекста устройства. EnumDisplayMonitors вызывает определяемую приложением функцию обратного вызова MonitorEnumProc один раз для каждого перечисленного монитора. Обратите внимание, что GetSystemMetrics (SM_CMONITORS) учитывает только мониторы дисплея.

[EnumDisplaySettingsA](#)

Функция EnumDisplaySettings извлекает сведения об одном из графических режимов для устройства отображения. Чтобы получить сведения для всех графических режимов устройства отображения, выполните ряд вызовов этой функции. (ANSI)

[EnumDisplaySettingsExA](#)

Функция EnumDisplaySettingsEx извлекает сведения об одном из графических режимов для устройства отображения. Чтобы получить сведения обо всех графических режимах для устройства отображения, выполните ряд вызовов этой функции. (ANSI)

[EnumDisplaySettingsExW](#)

Функция EnumDisplaySettingsEx извлекает сведения об одном из графических режимов для устройства отображения. Чтобы получить сведения обо всех графических режимах для устройства отображения, выполните ряд вызовов этой функции. (Юникод)

[EnumDisplaySettingsW](#)

Функция EnumDisplaySettings извлекает сведения об одном из графических режимов для устройства отображения. Чтобы получить сведения для всех графических режимов устройства отображения, выполните ряд вызовов этой функции. (Юникод)

[EnumPropsA](#)

Перечисляет все записи в списке свойств окна, передавая их по очереди в указанную функцию обратного вызова. EnumProps продолжается до перечисления последней записи или до тех пор, пока функция обратного вызова не вернет значение FALSE. (ANSI)

[EnumPropsExA](#)

Перечисляет все записи в списке свойств окна, передавая их по очереди в указанную функцию обратного вызова. EnumPropsEx продолжается до тех пор, пока не будет перечислена последняя запись, или функция обратного вызова не вернет значение FALSE. (ANSI)

[EnumPropsExW](#)

Перечисляет все записи в списке свойств окна, передавая их по очереди в указанную функцию обратного вызова. EnumPropsEx продолжается до тех пор, пока не будет перечислена последняя запись, или функция обратного вызова не вернет значение FALSE. (Юникод)

[EnumPropsW](#)

Перечисляет все записи в списке свойств окна, передавая их по очереди в указанную функцию обратного вызова. EnumProps продолжается до перечисления последней записи или до тех пор, пока функция обратного вызова не вернет значение FALSE. (Юникод)

[EnumThreadWindows](#)

Перечисляет все незашифрованные окна, связанные с потоком, передавая дескриптор каждому окну, в свою очередь, в определяемую приложением функцию обратного вызова.

[EnumWindows](#)

Перечисляет все окна верхнего уровня на экране, передавая дескриптор каждому окну, в свою очередь, в определяемую приложением функцию обратного вызова. EnumWindows продолжается до тех пор, пока не будет перечислено последнее окно верхнего уровня или функция обратного вызова не вернет значение FALSE.

[EnumWindowStationsA](#)

Перечисляет все оконные станции в текущем сеансе. Функция, в свою очередь, передает имя каждой оконной станции в определяемую приложением функцию обратного вызова. (ANSI)

[EnumWindowStationsW](#)

Перечисляет все оконные станции в текущем сеансе. Функция, в свою очередь, передает имя каждой оконной станции в определяемую приложением функцию обратного вызова. (Юникод)

[EqualRect](#)

Функция EqualRect определяет, равны ли два указанных прямоугольника, сравнивая координаты их верхнего левого и нижнего правых углов.

[EvaluateProximityToPolygon](#)

Возвращает оценку многоугольника в качестве вероятной цели касания (по сравнению со всеми другими многоугольниками, пересекающими контактную область касания), и скорректированную точку касания в многоугольнике.

[EvaluateProximityToRect](#)

Возвращает оценку прямоугольника в качестве вероятной цели касания по сравнению со всеми остальными прямоугольниками, пересекающими контактную область касания, и скорректированной точкой касания в прямоугольнике.

[ExcludeUpdateRgn](#)

Функция ExcludeUpdateRgn предотвращает рисование в недопустимых областях окна, исключая обновленную область в окне из обрезной области.

[ExitWindows](#)

Вызывает функцию ExitWindowsEx для выхода интерактивного пользователя.

[ExitWindowsEx](#)

Завершает работу интерактивного пользователя, завершает работу системы или завершает работу и перезапускает систему.

[FillRect](#)

Функция FillRect заполняет прямоугольник с помощью указанной кисти. Эта функция включает левую и верхнюю границы, но исключает правую и нижнюю границы прямоугольника.

[FindWindowA](#)

Извлекает дескриптор в окно верхнего уровня, имя класса и имя окна которого соответствуют указанным строкам. Эта функция не выполняет поиск дочерних окон. Эта функция не выполняет поиск с учетом регистра. (ANSI)

[FindWindowExA](#)

Извлекает дескриптор для окна, имя класса и имя окна которого совпадают с указанными строками. Функция выполняет поиск дочерних окон, начиная с следующего за указанным дочерним окном. Эта функция не выполняет поиск с учетом регистра. (ANSI)

[FindWindowExW](#)

Извлекает дескриптор для окна, имя класса и имя окна которого совпадают с указанными строками. Функция выполняет поиск дочерних окон, начиная с следующего за указанным дочерним окном. Эта функция не выполняет поиск с учетом регистра. (Юникод)

[FindWindowW](#)

Извлекает дескриптор в окно верхнего уровня, имя класса и имя окна которого соответствуют указанным строкам. Эта функция не выполняет поиск дочерних окон. Эта функция не выполняет поиск с учетом регистра. (Юникод)

[FlashWindow](#)

Мигает указанное окно один раз. Активное состояние окна не изменяется.

[FlashWindowEx](#)

Мигает указанное окно. Активное состояние окна не изменяется.

[FrameRect](#)

Функция FrameRect рисует границу вокруг указанного прямоугольника с помощью указанной кисти. Ширина и высота границы всегда являются одной логической единицей.

[GET_APPCOMMAND_LPARAM](#)

Извлекает команду приложения из указанного значения LPARAM.

[GET_DEVICE_LPARAM](#)

Извлекает тип устройства ввода из указанного значения LPARAM.

[GET_FLAGS_LPARAM](#)

Извлекает состояние определенных виртуальных клавиш из указанного значения LPARAM.
([GET_FLAGS_LPARAM](#))

[GET_KEYSTATE_LPARAM](#)

Извлекает состояние определенных виртуальных клавиш из указанного значения LPARAM.
([GET_KEYSTATE_LPARAM](#))

[GET_KEYSTATE_WPARAM](#)

Извлекает состояние определенных виртуальных клавиш из указанного значения WPARAM.

[GET_NCHITTEST_WPARAM](#)

Извлекает значение проверки попадания из указанного значения WPARAM.

[GET_POINTERID_WPARAM](#)

Извлекает идентификатор указателя, используя указанное значение.

[GET_RAWINPUT_CODE_WPARAM](#)

Извлекает входной код из wParam в WM_INPUT.

[GET_WHEEL_DELTA_WPARAM](#)

Извлекает значение wheel-delta из указанного значения WPARAM.

[GET_XBUTTON_WPARAM](#)

Извлекает состояние определенных кнопок из указанного значения WPARAM.

[GetActiveWindow](#)

Извлекает дескриптор окна в активное окно, присоединенное к очереди сообщений вызывающего потока.

[GetAltTabInfoA](#)

Извлекает сведения о состоянии указанного окна, если оно является окном переключения приложений (ALT+TAB). (ANSI)

[GetAltTabInfoW](#)

Извлекает сведения о состоянии указанного окна, если оно является окном переключения приложений (ALT+TAB). (Юникод)

[GetAncestor](#)

Извлекает дескриптор предку указанного окна.

[GetAsyncKeyState](#)

Определяет, находится ли клавиша вверх или вниз во время вызова функции и была ли клавиша нажата после предыдущего вызова GetAsyncKeyState.

[GetAutoRotationState](#)

Извлекает значение AR_STATE, содержащее состояние автоматического поворота экрана для системы, например, поддерживается ли автоматический поворот и включен ли он пользователем.

[GetAwarenessFromDpiAwarenessContext](#)

Извлекает значение DPI_AWARENESS из DPI_AWARENESS_CONTEXT.

[GetCapture](#)

Извлекает дескриптор в окно (если таковое имеется), которое захватило мышь. Только одно окно за раз может захватывать мышь; Это окно получает ввод с помощью мыши независимо от того, находится ли курсор в пределах его границ.

[GetCaretBlinkTime](#)

Извлекает время, необходимое для инвертировать пиксели курсора. Пользователь может задать это значение.

[GetCaretPos](#)

Копирует положение курсора в указанную структуру POINT.

[GetCIMSSM](#)

Извлекает источник входного сообщения (GetCurrentInputMessageSourceInSendMessage).

[GetClassInfoA](#)

Извлекает сведения о классе окна. (ANSI)

[GetClassInfoExA](#)

Извлекает сведения о классе окна, включая дескриптор небольшого значка, связанного с классом окна. Функция GetClassInfo не извлекает дескриптор маленького значка. (ANSI)

[GetClassInfoExW](#)

Извлекает сведения о классе окна, включая дескриптор небольшого значка, связанного с классом окна. Функция GetClassInfo не извлекает дескриптор маленького значка. (Юникод)

[GetClassInfoW](#)

Извлекает сведения о классе окна. (Юникод)

[GetClassLongA](#)

Извлекает указанное 32-разрядное значение (DWORD) из структуры WNDCLASSEX, связанной с указанным окном. (ANSI)

[GetClassLongPtrA](#)

Извлекает указанное значение из структуры WNDCLASSEX, связанной с указанным окном. (ANSI)

[GetClassLongPtrW](#)

Извлекает указанное значение из структуры WNDCLASSEX, связанной с указанным окном. (Юникод)

[GetClassLongW](#)

Извлекает указанное 32-разрядное значение (DWORD) из структуры WNDCLASSEX, связанной с указанным окном. (Юникод)

[GetClassName](#)

Функция GetClassName извлекает имя класса, которому принадлежит указанное окно. (GetClassName)

[GetClassNameA](#)

Извлекает имя класса, которому принадлежит указанное окно. (GetClassNameA)

[GetClassNameW](#)

Функция GetClassNameW (Юникод) извлекает имя класса, которому принадлежит указанное окно. (GetClassNameW)

[GetClassWord](#)

Извлекает 16-разрядное значение (WORD) с указанным смещением в дополнительную память класса для класса окна, которому принадлежит указанное окно.

[GetClientRect](#)

Извлекает координаты клиентской области окна.

[GetClipboardData](#)

Извлекает данные из буфера обмена в указанном формате. Буфер обмена должен быть открыт ранее.

[GetClipboardFormatNameA](#)

Извлекает из буфера обмена имя указанного зарегистрированного формата. Функция копирует имя в указанный буфер. (ANSI)

[GetClipboardFormatNameW](#)

Извлекает из буфера обмена имя указанного зарегистрированного формата. Функция копирует имя в указанный буфер. (Юникод)

[GetClipboardOwner](#)

Извлекает дескриптор окна текущего владельца буфера обмена.

[GetClipboardSequenceNumber](#)

Извлекает порядковый номер буфера обмена для текущей оконной станции.

[GetClipboardViewer](#)

Извлекает дескриптор в первое окно в цепочке просмотра буфера обмена.

[GetClipCursor](#)

Извлекает экранные координаты прямоугольной области, к которой ограничен курсор.

[GetComboBoxInfo](#)

Извлекает сведения о указанном поле со списком.

[GetCurrentInputMessageSource](#)

Извлекает источник входного сообщения.

[GetCursor](#)

Извлекает дескриптор для текущего курсора.

[GetCursorInfo](#)

Извлекает сведения о глобальном курсоре.

[GetCursorPos](#)

Извлекает положение курсора мыши в экранных координатах.

[GetDC](#)

Функция GetDC извлекает дескриптор в контекст устройства (DC) для клиентской области указанного окна или для всего экрана.

[GetDCEx](#)

Функция GetDCEx извлекает дескриптор в контекст устройства (DC) для клиентской области указанного окна или для всего экрана.

[GetDesktopWindow](#)

Извлекает дескриптор в окно рабочего стола. Окно рабочего стола охватывает весь экран. Окно рабочего стола — это область, поверх которой окрашены другие окна.

[GetDialogBaseUnits](#)

Извлекает базовые единицы диалогового окна системы, которые представляют собой среднюю ширину и высоту символов в системном шрифте.

[GetDialogControlDpiChangeBehavior](#)

Извлекает поведение масштабирования DPI для каждого монитора, переопределяющее дочернее окно в диалоговом окне.

[GetDialogDpiChangeBehavior](#)

Возвращает флаги, которые могли быть заданы в заданном диалоговом окне при предыдущем вызове Метода SetDialogDpiChangeBehavior.

[GetDisplayAutoRotationPreferences](#)

Извлекает настройки автоматического поворота экрана для текущего процесса.

[GetDisplayAutoRotationPreferencesByProcessId](#)

Извлекает параметры автоматического поворота экрана для процесса, указанного параметром dwProcessId.

[GetDisplayConfigBufferSizes](#)

Функция GetDisplayConfigBufferSizes извлекает размер буферов, необходимых для вызова функции QueryDisplayConfig.

[GetDlgCtrlID](#)

Извлекает идентификатор указанного элемента управления.

[GetDlgItem](#)

Извлекает дескриптор элемента управления в указанном диалоговом окне.

[GetDlgItemInt](#)

Преобразует текст указанного элемента управления в диалоговом окне в целочисленное значение.

[GetDlgItemTextA](#)

Извлекает заголовок или текст, связанный с элементом управления в диалоговом окне. (ANSI)

[GetDlgItemTextW](#)

Извлекает заголовок или текст, связанный с элементом управления в диалоговом окне. (Юникод)

[GetDoubleClickTime](#)

Извлекает текущее время двойного щелчка мыши.

[GetDpiAwarenessContextForProcess](#)

Возвращает дескриптор DPI_AWARENESS_CONTEXT для указанного процесса.

[GetDpiForSystem](#)

Возвращает системный DPI.

[GetDpiForWindow](#)

Возвращает значение точек на дюйм (dpi) для указанного окна.

[GetDpiFromDpiAwarenessContext](#)

Извлекает значение DPI из заданного дескриптора DPI_AWARENESS_CONTEXT. Это позволяет определить DPI потока без необходимости проверки окна, созданного в этом потоке.

[GetFocus](#)

Извлекает дескриптор для окна с фокусом клавиатуры, если окно подключено к очереди сообщений вызывающего потока.

[GetForegroundWindow](#)

Извлекает дескриптор для окна переднего плана (окна, с которым в данный момент работает пользователь). Система назначает потоку, который создает окно переднего плана, немного более высокий приоритет, чем другим потокам.

[GetGestureConfig](#)

Извлекает конфигурацию, для которой сообщения жестов Windows Touch отправляются из окна.

[GetGestureExtraArgs](#)

Извлекает дополнительные сведения о жесте из дескриптора GESTUREINFO.

[GetGestureInfo](#)

Извлекает структуру GESTUREINFO, заданную дескриптором для сведений жеста.

[GetGuiResources](#)

Извлекает количество дескрипторов объектов графического пользовательского интерфейса (GUI), используемых указанным процессом.

[GetGUIThreadInfo](#)

Извлекает сведения об активном окне или указанном потоке графического пользовательского интерфейса.

[GetIconInfo](#)

Извлекает сведения об указанном значке или курсоре.

[GetIconInfoExA](#)

Извлекает сведения об указанном значке или курсоре. GetIconInfoEx расширяет GetIconInfo с помощью новой структуры ICONINFOEX. (ANSI)

[GetIconInfoExW](#)

Извлекает сведения об указанном значке или курсоре. GetIconInfoEx расширяет GetIconInfo с помощью новой структуры ICONINFOEX. (Юникод)

[GetInputState](#)

Определяет, есть ли в очереди сообщений вызывающего потока сообщения с помощью кнопки мыши или клавиатуры.

[GetKBCodePage](#)

Извлекает текущую кодовую страницу.

[GetKeyboardLayout](#)

Извлекает идентификатор активного языкового стандарта ввода (прежнее название — раскладка клавиатуры).

[GetKeyboardLayoutList](#)

Извлекает идентификаторы входных языковых стандартов (ранее называемые дескрипторами раскладки клавиатуры), соответствующие текущему набору языковых стандартов ввода в системе. Функция копирует идентификаторы в указанный буфер.

[GetKeyboardLayoutNameA](#)

Извлекает имя идентификатора активного языкового стандарта ввода (ранее называвшийся раскладкой клавиатуры) для системы. (ANSI)

[GetKeyboardLayoutNameW](#)

Извлекает имя идентификатора активного языкового стандарта ввода (ранее называвшийся раскладкой клавиатуры) для системы. (Юникод)

[GetKeyboardState](#)

Копирует состояние 256 виртуальных ключей в указанный буфер.

[GetKeyboardType](#)

Извлекает сведения о текущей клавиатуре.

[GetKeyNameTextA](#)

Извлекает строку, представляющую имя ключа. (ANSI)

[GetKeyNameTextW](#)

Извлекает строку, представляющую имя ключа. (Юникод)

[GetKeyState](#)

Возвращает состояние указанного виртуального ключа. Состояние указывает, находится ли клавиша вверх, вниз или переключается (включено, отключено при каждом нажатии клавиши).

[GetLastActivePopup](#)

Определяет, какое всплывающее окно, принадлежавшее указанному окну, было последнее активное.

[GetLastInputInfo](#)

Извлекает время последнего входного события.

[GetLayeredWindowAttributes](#)

Получает ключ цвета прозрачности многослойного окна.

[GetListBoxInfo](#)

Извлекает количество элементов на столбец в указанном списке.

[GetMenu](#)

Извлекает дескриптор для меню, назначенного указанному окну.

[GetMenuBarInfo](#)

Извлекает сведения об указанной строке меню.

[GetMenuCheckMarkDimensions](#)

Извлекает измерения растрового изображения проверка метки по умолчанию.

[GetMenuContextHelpId](#)

Извлекает идентификатор контекста справки, связанный с указанным меню.

[GetMenuItemDefault](#)

Определяет пункт меню по умолчанию в указанном меню.

[GetMenuItemInfo](#)

Извлекает сведения об указанном меню.

[GetMenuItemCount](#)

Определяет количество элементов в указанном меню.

[GetMenuItemID](#)

Извлекает идентификатор элемента меню, расположенного в указанной позиции в меню.

[GetMenuItemInfoA](#)

Извлекает сведения об элементе меню. (ANSI)

[GetMenuItemInfoW](#)

Извлекает сведения об элементе меню. (Юникод)

[GetMenuItemRect](#)

Извлекает ограничивающий прямоугольник для указанного пункта меню.

[GetMenuState](#)

Извлекает флаги меню, связанные с указанным элементом меню.

[GetMenuItemStringA](#)

Копирует текстовую строку указанного пункта меню в указанный буфер. (ANSI)

[GetMenuItemStringW](#)

Копирует текстовую строку указанного пункта меню в указанный буфер. (Юникод)

[GetMessage](#)

Функция GetMessage извлекает сообщение из очереди сообщений вызывающего потока. (GetMessage)

[GetMessageA](#)

Извлекает сообщение из очереди сообщений вызывающего потока. Функция отправляет входящие отправленные сообщения до тех пор, пока отправленное сообщение не будет доступно для извлечения. (GetMessageA)

[GetMessageExtraInfo](#)

Извлекает дополнительные сведения о сообщении для текущего потока. Дополнительные сведения о сообщении — это определяемое приложением или драйвером значение, связанное с очередью сообщений текущего потока.

[GetMessagePos](#)

Извлекает позицию курсора для последнего сообщения, полученного функцией GetMessage.

[GetMessageTime](#)

Извлекает время последнего сообщения, полученного функцией GetMessage.

[GetMessageW](#)

Функция GetMessageW (Юникод) извлекает сообщение из очереди сообщений вызывающего потока. (GetMessageW)

[GetMonitorInfoA](#)

Функция GetMonitorInfo извлекает сведения о мониторе дисплея. (ANSI)

[GetMonitorInfoW](#)

Функция GetMonitorInfo извлекает сведения о мониторе дисплея. (Юникод)

[GetMouseMovePointsEx](#)

Извлекает журнал до 64 предыдущих координат мыши или пера.

[GetNextDlgGroupItem](#)

Извлекает дескриптор первого элемента управления в группе элементов управления, которые предшествуют указанному элементу управления в диалоговом окне или следуют за ней.

[GetNextDlgTabItem](#)

Извлекает дескриптор первого элемента управления, который имеет стиль WS_TABSTOP, который предшествует указанному элементу управления (или следует за ней).

[GetNextWindow](#)

Извлекает дескриптор следующего или предыдущего окна в Z-порядке. Следующее окно находится под указанным окном; предыдущее окно находится выше.

[GetOpenClipboardWindow](#)

Извлекает дескриптор в окно, в которое в настоящее время открыт буфер обмена.

[GetParent](#)

Извлекает дескриптор родительского или владельца указанного окна.

[GetPhysicalCursorPos](#)

Извлекает положение курсора в физических координатах.

[GetPointerCursorId](#)

Извлекает идентификатор курсора, связанный с указанным указателем.

[GetPointerDevice](#)

Возвращает сведения об устройстве указателя.

[GetPointerDeviceCursors](#)

Получает идентификаторы курсоров, сопоставленные с курсорами, связанными с устройством указателя.

[GetPointerDeviceProperties](#)

Возвращает свойства устройства, которые не включены в структуру `PINTER_DEVICE_INFO`.

[GetPointerDeviceRects](#)

Возвращает диапазоны x и y для устройства указателя (в himetric) и диапазоны x и y (текущее разрешение) для дисплея, с которым сопоставлено устройство указателя.

[GetPointerDevices](#)

Возвращает сведения об устройствах-указателях, подключенных к системе.

[GetPointerFrameInfo](#)

Возвращает весь кадр сведений для указанных указателей, связанных с текущим сообщением.

[GetPointerFrameInfoHistory](#)

Возвращает весь кадр данных (включая объединяемые входные кадры) для указанных указателей, связанных с текущим сообщением.

[GetPointerFramePenInfo](#)

Возвращает весь кадр сведений на основе пера для указанных указателей (типа PT_PEN), связанных с текущим сообщением.

[GetPointerFramePenInfoHistory](#)

Возвращает весь кадр сведений на основе пера (включая объединяемые входные кадры) для указанных указателей (типа PT_PEN), связанных с текущим сообщением.

[GetPointerFrameTouchInfo](#)

Возвращает весь кадр сенсорной информации для указанных указателей (типа PT_TOUCH), связанных с текущим сообщением.

[GetPointerFrameTouchInfoHistory](#)

Возвращает весь кадр сенсорной информации (включая объединяемые входные кадры) для указанных указателей (типа PT_TOUCH), связанных с текущим сообщением.

[GetPointerInfo](#)

Возвращает сведения для указанного указателя, связанного с текущим сообщением.

[GetPointerInfoHistory](#)

Возвращает сведения, связанные с отдельными входными данными , если таковые имелись, которые были сгруппированы в текущее сообщение для указанного указателя.

[GetPointerInputTransform](#)

Возвращает одно или несколько преобразований для координат сведений о указателе, связанных с текущим сообщением.

[GetPointerPenInfo](#)

Возвращает сведения на основе пера для указанного указателя (типа PT_PEN), связанного с текущим сообщением.

[GetPointerPenInfoHistory](#)

Возвращает сведения на основе пера, связанные с отдельными входными данными , если таковые были, которые были объединялись в текущее сообщение для указанного указателя (типа PT_PEN).

[GetPointerTouchInfo](#)

Возвращает сведения на основе касания для указанного указателя (типа PT_TOUCH), связанного с текущим сообщением.

[GetPointerTouchInfoHistory](#)

Возвращает информацию на основе сенсорного ввода, связанную с отдельными входными данными , если таковые имелись, которые были сгруппированы в текущее сообщение для указанного указателя (типа PT_TOUCH).

[GetPointerType](#)

Извлекает тип указателя для указанного указателя.

[GetPriorityClipboardFormat](#)

Извлекает первый доступный формат буфера обмена в указанном списке.

[GetProcessDefaultLayout](#)

Извлекает макет по умолчанию, используемый при создании окон без родительского или владельца.

[GetProcessWindowStation](#)

Извлекает дескриптор текущей оконной станции для вызывающего процесса.

[GetProp](#)

Извлекает дескриптор данных из списка свойств указанного окна. Стока символов идентифицирует извлекаемую дескриптор. Стока и дескриптор должны быть добавлены в список свойств при предыдущем вызове функции SetProp. (ANSI)

[GetPropW](#)

Извлекает дескриптор данных из списка свойств указанного окна. Стока символов идентифицирует извлекаемую дескриптор. Стока и дескриптор должны быть добавлены в список свойств при предыдущем вызове функции SetProp. (Юникод)

[GetQueueStatus](#)

Извлекает тип сообщений, найденных в очереди сообщений вызывающего потока.

[GetRawInputBuffer](#)

Выполняет буферизованное чтение необработанных входных данных.

[GetRawInputData](#)

Извлекает необработанные входные данные с указанного устройства.

[GetRawInputDeviceInfoA](#)

Извлекает сведения о необработанном устройстве ввода. (ANSI)

[GetRawInputDeviceInfoW](#)

Извлекает сведения о необработанном устройстве ввода. (Юникод)

[GetRawInputDeviceList](#)

Перечисляет необработанные устройства ввода, подключенные к системе.

[GetRawPointerDeviceData](#)

Возвращает необработанные входные данные с устройства указателя.

[GetRegisteredRawInputDevices](#)

Извлекает сведения о необработанных устройствах ввода для текущего приложения.

[GetScrollBarInfo](#)

Функция GetScrollBarInfo извлекает сведения об указанной полосе прокрутки.

[GetScrollInfo](#)

Функция GetScrollInfo извлекает параметры полосы прокрутки, включая минимальную и максимальную позиции прокрутки, размер страницы и положение поля прокрутки (большого пальца).

[GetScrollPos](#)

Функция GetScrollPos извлекает текущее положение поля прокрутки (большого пальца) на указанной полосе прокрутки.

[GetScrollRange](#)

Функция GetScrollRange извлекает текущую минимальную и максимальную позиции поля прокрутки (большого пальца) для указанной полосы прокрутки.

[GetShellWindow](#)

Извлекает дескриптор в окно рабочего стола оболочки.

[GetSubMenu](#)

Извлекает дескриптор раскрывающегося меню или подменю, активированного указанным элементом меню.

[GetSysColor](#)

Извлекает текущий цвет указанного отображаемого элемента.

[GetSysColorBrush](#)

Функция GetSysColorBrush извлекает дескриптор, определяющий логическую кисть, соответствующую указанному индексу цвета.

[GetSystemDpiForProcess](#)

Извлекает системный DPI, связанный с заданным процессом. Это полезно для предотвращения проблем совместимости, возникающих в связи с обменом конфиденциальными данными DPI между несколькими системными процессами с разными системными значениями DPI.

[GetSystemMenu](#)

Позволяет приложению получить доступ к меню окна (также известному как системное меню или меню управления) для копирования и изменения.

[GetSystemMetrics](#)

Извлекает указанную системную метрику или параметр конфигурации системы.

[GetSystemMetricsForDpi](#)

Извлекает указанную системную метрику или параметр конфигурации системы с учетом указанного DPI.

[GetTabbedTextExtentA](#)

Функция GetTabbedTextExtent вычисляет ширину и высоту строки символов. (ANSI)

[GetTabbedTextExtentW](#)

Функция GetTabbedTextExtent вычисляет ширину и высоту строки символов. (Юникод)

[GetThreadDesktop](#)

Извлекает дескриптор на рабочий стол, назначенный указанному потоку.

[GetThreadDpiAwarenessContext](#)

Возвращает DPI_AWARENESS_CONTEXT для текущего потока.

[GetThreadDpiHostingBehavior](#)

Извлекает DPI_HOSTING_BEHAVIOR из текущего потока.

[GetTitleBarInfo](#)

Извлекает сведения об указанной строке заголовка.

[GetTopWindow](#)

Проверяет порядок Z дочерних окон, связанных с указанным родительским окном, и извлекает дескриптор дочернего окна в верхней части порядка Z.

[GetTouchInputInfo](#)

Извлекает подробные сведения о сенсорных входных данных, связанных с определенным дескриптором сенсорного ввода.

[GetUnpredictedMessagePos](#)

Получает данные указателя перед обработкой сенсорного прогнозирования.

[GetUpdatedClipboardFormats](#)

Извлекает поддерживаемые в настоящее время форматы буфера обмена.

[GetUpdateRect](#)

Функция GetUpdateRect извлекает координаты наименьшего прямоугольника, полностью включающего область обновления указанного окна.

[GetUpdateRgn](#)

Функция GetUpdateRgn извлекает область обновления окна, копируя ее в указанную область. Координаты области обновления находятся относительно левого верхнего угла окна (то есть являются клиентскими координатами).

[GetUserObjectInformationA](#)

Извлекает сведения об указанной оконной станции или объекте рабочего стола. (ANSI)

[GetUserObjectInformationW](#)

Извлекает сведения об указанной оконной станции или объекте рабочего стола. (Юникод)

[GetUserObjectSecurity](#)

Извлекает сведения о безопасности для указанного объекта пользователя.

[GetWindow](#)

Извлекает дескриптор для окна, которое имеет указанную связь (Z-порядок или владелец) с указанным окном.

[GetWindowContextHelpId](#)

Извлекает идентификатор контекста справки, если таковой есть, связанный с указанным окном.

[GetWindowDC](#)

Функция GetWindowDC извлекает контекст устройства (DC) для всего окна, включая заголовок окна, меню и полосы прокрутки.

[GetWindowDisplayAffinity](#)

Извлекает текущий параметр сходства отображения из любого процесса для заданного окна.

[GetWindowDpiAwarenessContext](#)

Возвращает DPI_AWARENESS_CONTEXT, связанный с окном.

[GetWindowDpiHostingBehavior](#)

Возвращает DPI_HOSTING_BEHAVIOR указанного окна.

[GetWindowFeedbackSetting](#)

Извлекает конфигурацию обратной связи для окна.

[GetWindowInfo](#)

Извлекает сведения об указанном окне. (GetWindowInfo)

[GetWindowLongA](#)

Извлекает сведения об указанном окне. (GetWindowLongA)

[GetWindowLongPtrA](#)

Извлекает сведения об указанном окне. Функция также извлекает значение с указанным смещением в дополнительную память окна. (ANSI)

[GetWindowLongPtrW](#)

Извлекает сведения об указанном окне. Функция также извлекает значение с указанным смещением в дополнительную память окна. (Юникод)

[GetWindowLongW](#)

Извлекает сведения об указанном окне. (GetWindowLongW)

[GetWindowModuleFileNameA](#)

Извлекает полный путь и имя файла модуля, связанного с указанным дескриптором окна.
(ANSI)

[GetWindowModuleFileNameW](#)

Извлекает полный путь и имя файла модуля, связанного с указанным дескриптором окна.
(Юникод)

[GetWindowPlacement](#)

Извлекает состояние отображения и восстановленные, свернутые и развернутые позиции
указанного окна.

[GetWindowRect](#)

Извлекает размеры ограничивающего прямоугольника указанного окна. Измерения
задаются в координатах экрана, относящихся к левому верхнему углу экрана.

[GetWindowRgn](#)

Функция GetWindowRgn получает копию области окна.

[GetWindowRgnBox](#)

Функция GetWindowRgnBox извлекает размеры самого жесткого ограничивающего
прямоугольника для области окна.

[GetWindowTextA](#)

Копирует текст строки заголовка указанного окна (если она имеется) в буфер. Если
указанное окно является элементом управления, копируется текст элемента управления .
Однако GetWindowText не может получить текст элемента управления в другом приложении.
(ANSI)

[GetWindowTextLengthA](#)

Извлекает длину (в символах) текста заголовка указанного окна (если окно имеет строку
заголовка). (ANSI)

[GetWindowTextLengthW](#)

Извлекает длину (в символах) текста заголовка указанного окна (если окно имеет строку
заголовка). (Юникод)

[GetWindowTextW](#)

Копирует текст строки заголовка указанного окна (если она имеется) в буфер. Если указанное окно является элементом управления, копируется текст элемента управления . Однако GetWindowText не может получить текст элемента управления в другом приложении. (Юникод)

[GetWindowThreadProcessId](#)

Извлекает идентификатор потока, создавшего указанное окно, и, при необходимости, идентификатор процесса, создавшего окно.

[GetWindowWord](#)

Извлекает 16-битовое значение (**DWORD**) с указанным смещением в дополнительное окно памяти.

[GID_ROTATE_ANGLE_FROM_ARGUMENT](#)

Макрос GID_ROTATE_ANGLE_FROM_ARGUMENT используется для интерпретации значения ullArgument GID_ROTATE при получении значения в структуре WM_GESTURE.

[GID_ROTATE_ANGLE_TO_ARGUMENT](#)

Преобразует радиановое значение в аргумент для сообщений жеста поворота.

[GrayStringA](#)

Функция GrayString рисует серый текст в указанном расположении. (ANSI)

[GrayStringW](#)

Функция GrayString рисует серый текст в указанном расположении. (Юникод)

[HAS_POINTER_CONFIDENCE_WPARAM](#)

Проверяет, считается ли указанное сообщение указателя преднамеренным, а не случайным.

[HideCaret](#)

Удаляет курсор с экрана. Скрытие курсора не приводит к уничтожению текущей фигуры и не делает ее недействительной.

[HiliteMenuItem](#)

Добавляет или удаляет выделение из элемента в строке меню.

[InflateRect](#)

Функция InflateRect увеличивает или уменьшает ширину и высоту указанного прямоугольника.

[InheritWindowMonitor](#)

Приводит к тому, что указанное окно наследует монитор другого окна.

[InitializeTouchInjection](#)

Настраивает контекст внедрения сенсорного ввода для вызывающего приложения и инициализирует максимальное количество одновременных контактов, которые приложение может внедрять.

[InjectSyntheticPointerInput](#)

Имитирует ввод указателя (перо или сенсорный ввод).

[InjectTouchInput](#)

Имитирует сенсорный ввод.

[InSendMessage](#)

Определяет, обрабатывает ли текущая процедура окна сообщение, отправленное из другого потока (в том же или другом процессе) путем вызова функции SendMessage.

[InSendMessageEx](#)

Определяет, обрабатывает ли текущая процедура окно сообщение, отправленное из другого потока (в том же или другом процессе).

[InsertMenuItemA](#)

Вставляет новый пункт меню в меню, перемещая другие элементы вниз по меню. (ANSI)

[InsertMenuItemW](#)

Вставляет новый пункт меню в указанную позицию в меню. (Юникод)

[InsertMenuW](#)

Вставляет новый пункт меню в меню, перемещая другие элементы вниз по меню. (Юникод)

[InternalGetWindowText](#)

Копирует текст строки заголовка указанного окна (если она имеется) в буфер.

[IntersectRect](#)

Функция `IntersectRect` вычисляет пересечение двух исходных прямоугольников и помещает координаты прямоугольника пересечения в прямоугольник назначения.

[InvalidateRect](#)

Функция `InvalidateRect` добавляет прямоугольник в область обновления указанного окна. Область обновления представляет часть клиентской области окна, которую необходимо перерисовать.

[InvalidateRgn](#)

Функция `InvalidateRgn` делает недействительной клиентную область в указанном регионе, добавляя ее в текущую область обновления окна.

[Инвертирование](#)

Функция `InvertRect` инвертирует прямоугольник в окне, выполняя логическую операцию NOT со значениями цвета для каждого пикселя внутри прямоугольника.

[IS_INTRESOURCE](#)

Определяет, является ли значение целочисленным идентификатором ресурса.

[IS_POINTER_CANCELED_WPARAM](#)

Проверяет, завершился ли указанный ввод указателя внезапно или был ли он недопустимым, указывая на то, что взаимодействие не было завершено.

[IS_POINTER_FIFTHBUTTON_WPARAM](#)

Проверяет, выполнил ли указанный указатель пятое действие.

[IS_POINTER_FIRSTBUTTON_WPARAM](#)

Проверяет, выполнил ли указанный указатель первое действие.

[IS_POINTER_FLAG_SET_WPARAM](#)

Проверяет, задает ли макрос указателя указанный флаг.

[IS_POINTER_FOURTHBUTTON_WPARAM](#)

Проверяет, выполнил ли указанный указатель четвертое действие.

[IS_POINTER_INCONTACT_WPARAM](#)

Проверяет, находится ли указанный указатель в контакте.

[IS_POINTER_INRANGE_WPARAM](#)

Проверяет, находится ли указанный указатель в диапазоне.

[IS_POINTER_NEW_WPARAM](#)

Проверяет, является ли указанный указатель новым указателем.

[IS_POINTER_SECONDBUTTON_WPARAM](#)

Проверяет, выполнил ли указанный указатель второе действие.

[IS_POINTER_THIRDBUTTON_WPARAM](#)

Проверяет, выполнил ли указанный указатель третье действие.

[IsCharAlphaA](#)

Определяет, является ли символ символом в алфавитном порядке. Это определение основано на семантике языка, выбранного пользователем во время установки или с помощью панель управления. (ANSI)

[IsCharAlphaNumericA](#)

Определяет, является ли символ алфавитным или числовым символом. Это определение основано на семантике языка, выбранного пользователем во время установки или с помощью панель управления. (ANSI)

[IsCharAlphaNumericW](#)

Определяет, является ли символ алфавитным или числовым символом. Это определение основано на семантике языка, выбранного пользователем во время установки или с помощью панель управления. (Юникод)

[IsCharAlphaw](#)

Определяет, является ли символ символом в алфавитном порядке. Это определение основано на семантике языка, выбранного пользователем во время установки или с помощью панель управления. (Юникод)

[IsCharLowerA](#)

Определяет, является ли символ строчным регистром. Это определение основано на семантике языка, выбранного пользователем во время установки или с помощью панель управления.

[IsCharLowerW](#)

Функция `IsCharLowerW` (Юникод) определяет, является ли символ строчным. (`IsCharLowerW`)

[IsCharUpperA](#)

Определяет, является ли символ прописным. Это определение основано на семантике языка, выбранного пользователем во время установки или с помощью панель управления. (ANSI)

[IsCharUpperW](#)

Определяет, является ли символ прописным. Это определение основано на семантике языка, выбранного пользователем во время установки или с помощью панель управления. (Юникод)

[IsChild](#)

Определяет, является ли окно дочерним окном или окном-потомком указанного родительского окна.

[IsClipboardFormatAvailable](#)

Определяет, содержит ли буфер обмена данные в указанном формате.

[IsDialogMessageA](#)

Определяет, предназначено ли сообщение для указанного диалогового окна, и, если оно имеется, обрабатывает сообщение. (ANSI)

[IsDialogMessageW](#)

Определяет, предназначено ли сообщение для указанного диалогового окна, и, если оно имеется, обрабатывает сообщение. (Юникод)

[IsDlgButtonChecked](#)

Функция `IsDlgButtonChecked` определяет, установлен ли флажок кнопки или установлен флажок с тремя состояниями.

[IsGUIThread](#)

Определяет, является ли вызывающий поток потоком графического пользовательского интерфейса. Кроме того, при необходимости можно преобразовать поток в поток графического пользовательского интерфейса.

[IsHungAppWindow](#)

Определяет, считает ли система, что указанное приложение не отвечает.

IsIconic	Определяет, свернуто ли указанное окно (знаковые знаки).
IsImmersiveProcess	Определяет, принадлежит ли процесс приложению Магазина Windows.
IsMenu	Определяет, является ли дескриптор дескриптором меню.
IsMouseInPointerEnabled	Указывает, настроен ли параметр EnableMouseInPointer для мыши в качестве устройства ввода указателя и отправки WM_POINTER сообщений.
IsProcessDPIAware	IsProcessDPIAware может быть изменено или недоступно. Вместо этого используйте GetProcessDPIAwareness.
IsRectEmpty	Функция IsRectEmpty определяет, является ли указанный прямоугольник пустым.
IsTouchWindow	Проверяет, поддерживает ли указанное окно сенсорное управление, и при необходимости получает флаги модификаторов, заданные для сенсорной возможности окна.
IsValidDpiAwarenessContext	Определяет, является ли указанное DPI_AWARENESS_CONTEXT допустимым и поддерживаемым текущей системой.
IsWindow	Определяет, определяет ли указанный дескриптор окна существующее окно.
IsWindowArranged	Определяет, упорядочено ли указанное окно (то есть прикреплено ли оно).
IsWindowEnabled	Определяет, включено ли указанное окно для ввода с помощью мыши и клавиатуры.

[IsWindowUnicode](#)

Определяет, является ли указанное окно собственным окном Юникода.

[IsWindowVisible](#)

Определяет состояние видимости указанного окна.

[IsWinEventHookInstalled](#)

Определяет, есть ли установленный перехватчик WinEvent, который может быть уведомлен о указанном событии.

[IsWow64Message](#)

Определяет, получено ли последнее сообщение из очереди текущего потока из процесса WOW64.

[IsZoomed](#)

Определяет, развернуто ли окно.

[keybd_event](#)

Синтезирует нажатие клавиши.

[KillTimer](#)

Уничтожает указанный таймер.

[LoadAcceleratorsA](#)

Загружает указанную таблицу ускорителей. (ANSI)

[LoadAcceleratorsW](#)

Загружает указанную таблицу ускорителей. (Юникод)

[LoadBitmapA](#)

Функция LoadBitmap загружает указанный ресурс растрового изображения из исполняемого файла модуля. (ANSI)

[LoadBitmapW](#)

Функция LoadBitmap загружает указанный ресурс растрового изображения из исполняемого файла модуля. (Юникод)

[LoadCursorA](#)

Загружает указанный ресурс курсора из исполняемого (.EXE) файла, связанного с экземпляром приложения. (ANSI)

[LoadCursorFromFileA](#)

Создает курсор на основе данных, содержащихся в файле. (ANSI)

[LoadCursorFromFileW](#)

Создает курсор на основе данных, содержащихся в файле. (Юникод)

[LoadCursorW](#)

Загружает указанный ресурс курсора из исполняемого (.EXE) файла, связанного с экземпляром приложения. (Юникод)

[LoadIconA](#)

Загружает указанный ресурс значка из исполняемого файла (.exe), связанного с экземпляром приложения. (ANSI)

[LoadIconW](#)

Загружает указанный ресурс значка из исполняемого файла (.exe), связанного с экземпляром приложения. (Юникод)

[LoadImageA](#)

Загружает значок, курсор, анимированный курсор или точечный рисунок. (ANSI)

[LoadImageW](#)

Загружает значок, курсор, анимированный курсор или точечный рисунок. (Юникод)

[LoadKeyboardLayoutA](#)

Загружает в систему новый идентификатор языкового стандарта ввода (прежнее название — раскладка клавиатуры). (ANSI)

[LoadKeyboardLayoutW](#)

Загружает в систему новый идентификатор языкового стандарта ввода (прежнее название — раскладка клавиатуры). (Юникод)

[LoadMenuA](#)

Загружает указанный ресурс меню из исполняемого файла (.exe), связанного с экземпляром приложения. (ANSI)

[LoadMenuIndirectA](#)

Загружает указанный шаблон меню в память. (ANSI)

[LoadMenuIndirectW](#)

Загружает указанный шаблон меню в память. (Юникод)

[LoadMenuW](#)

Загружает указанный ресурс меню из исполняемого файла (.exe), связанного с экземпляром приложения. (Юникод)

[LoadStringA](#)

Загружает строковый ресурс из исполняемого файла, связанного с указанным модулем, копирует строку в буфер и добавляет завершающий символ NULL. (ANSI)

[LoadStringW](#)

Загружает строковый ресурс из исполняемого файла, связанного с указанным модулем, копирует строку в буфер и добавляет завершающий символ NULL. (Юникод)

[LockSetForegroundWindow](#)

Процесс переднего плана может вызвать функцию LockSetForegroundWindow, чтобы отключить вызовы функции SetForegroundWindow.

[LockWindowUpdate](#)

Функция LockWindowUpdate отключает или включает рисование в указанном окне. Одновременно можно заблокировать только одно окно.

[LockWorkStation](#)

Блокирует дисплей рабочей станции.

[LogicalToPhysicalPoint](#)

Преобразует логические координаты точки в окне в физические.

[LogicalToPhysicalPointForPerMonitorDPI](#)

Преобразует точку в окне из логических координат в физические, независимо от того, какие точки на дюйм (точек на дюйм) учитывает вызывающий объект.

[LookupIconIdFromDirectory](#)

Выполняет поиск значка или курсора по данным значка или курсора, наиболее подходящих для текущего устройства отображения. ([LookupIconIdFromDirectory](#))

[LookupIconIdFromDirectoryEx](#)

Выполняет поиск значка или курсора по данным значка или курсора, наиболее подходящих для текущего устройства отображения. ([LookupIconIdFromDirectoryEx](#))

[MAKEINTRESOURCEA](#)

Преобразует целочисленное значение в тип ресурса, совместимый с функциями управления ресурсами. Этот макрос используется вместо строки, содержащей имя ресурса. (ANSI)

[MAKEINTRESOURCEW](#)

Преобразует целочисленное значение в тип ресурса, совместимый с функциями управления ресурсами. Этот макрос используется вместо строки, содержащей имя ресурса. (Юникод)

[MAKELPARAM](#)

Создает значение для использования в качестве параметра lParam в сообщении. Макрос объединяет указанные значения.

[MAKELRESULT](#)

Создает значение для использования в качестве возвращаемого значения из процедуры окна. Макрос объединяет указанные значения.

[MAKEWPARAM](#)

Создает значение для использования в качестве параметра wParam в сообщении. Макрос объединяет указанные значения.

[MapDialogRect](#)

Преобразует указанные единицы диалогового окна в единицы экрана (пиксели).

[MapVirtualKeyA](#)

Преобразует (сопоставляет) код виртуального ключа в код сканирования или значение символа или преобразует код сканирования в код виртуального ключа. (ANSI)

[MapVirtualKeyExA](#)

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. Функция преобразует коды с помощью языка ввода и идентификатора языкового стандарта. (ANSI)

[MapVirtualKeyExW](#)

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. Функция преобразует коды с помощью языка ввода и идентификатора языкового стандарта. (Юникод)

[MapVirtualKeyW](#)

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. (Юникод)

[MapWindowPoints](#)

Функция MapWindowPoints преобразует (сопоставляет) набор точек из координатного пространства относительно одного окна в координатное пространство относительно другого окна.

[MenuItemFromPoint](#)

Определяет, какой пункт меню, если таковой имеется, находится в указанном расположении.

[MessageBeep](#)

Воспроизводит звуковой сигнал. Звуковой сигнал для каждого типа звука определяется записью в реестре.

[Messagebox](#)

Функция MessageBox отображает модальное диалоговое окно, содержащее системный значок, набор кнопок и краткое сообщение для конкретного приложения.

[MessageBoxA](#)

Отображает модальное диалоговое окно, содержащее системный значок, набор кнопок и краткое сообщение для конкретного приложения, например сведения о состоянии или ошибке. Окно сообщения возвращает целочисленное значение, указывающее, какую кнопку нажал пользователь. (MessageBoxA)

[MessageBoxExA](#)

Создает, отображает и управляет окном сообщения. (ANSI)

[MessageBoxExW](#)

Создает, отображает и управляет окном сообщения. (Юникод)

[MessageBoxIndirectA](#)

Создает, отображает и управляет окном сообщения. Окно сообщения содержит текст и заголовок сообщения, определяемые приложением, любой значок и любое сочетание предопределенных кнопок. (ANSI)

[MessageBoxIndirectW](#)

Создает, отображает и управляет окном сообщения. Окно сообщения содержит текст и заголовок сообщения, определяемые приложением, любой значок и любое сочетание предопределенных кнопок. (Юникод)

[MessageBoxW](#)

Функция MessageBoxW (Юникод) отображает модальное диалоговое окно, содержащее системный значок, набор кнопок и краткое сообщение для конкретного приложения.

[ModifyMenuA](#)

Изменяет существующий пункт меню. (ANSI)

[ModifyMenuW](#)

Изменяет существующий пункт меню. (Юникод)

[MonitorFromPoint](#)

Функция MonitorFromPoint извлекает дескриптор монитора дисплея, который содержит указанную точку.

[MonitorFromRect](#)

Функция MonitorFromRect извлекает дескриптор монитора дисплея, который имеет наибольшую область пересечения с указанным прямоугольником.

[MonitorFromWindow](#)

Функция MonitorFromWindow извлекает дескриптор монитора дисплея, который имеет наибольшую область пересечения с ограничивающим прямоугольником указанного окна.

[mouse_event](#)

Функция `mouse_event` синтезирует движения мыши и нажатия кнопок.

[MoveWindow](#)

Изменяет положение и размеры указанного окна.

[MsgWaitForMultipleObjects](#)

Ожидает, пока один или все указанные объекты не будут в состоянии сигнала или не истечет интервал времени ожидания. Объекты могут включать объекты входных событий.

[MsgWaitForMultipleObjectsEx](#)

Ожидает, пока один или все указанные объекты не будут помещены в состояние сигнала, подпрограмма завершения ввода-вывода или асинхронный вызов процедуры (APC) будет помещен в очередь в поток или истекло время ожидания. Массив объектов может включать объекты входных событий.

[NEXTRAWINPUTBLOCK](#)

Извлекает расположение следующей структуры в массиве структур RAWINPUT.

[NotifyWinEvent](#)

Сообщает системе, что произошло предопределенное событие. Если какие-либо клиентские приложения зарегистрировали функцию перехватчика для события, система вызывает функцию перехватчика клиента.

[OemKeyScan](#)

Сопоставляет коды OEMASCII от 0 до 0x0FF с кодами сканирования OEM и состояниями сдвига. Функция предоставляет сведения, позволяющие программе отправлять текст OEM в другую программу путем имитации ввода с клавиатуры.

[OemToCharA](#)

Преобразует строку из набора символов, определяемого изготовителем оборудования, в ansi или строку расширенных символов. Предупреждение Не используйте. (ANSI)

[OemToCharBuffA](#)

Преобразует указанное число символов в строке из набора символов, определяемого OEM, в anSI или строку расширенных символов. (ANSI)

[OemToCharBuffW](#)

Преобразует указанное число символов в строке из набора символов, определяемого OEM, в anSI или строку расширенных символов. (Юникод)

[OemToCharW](#)

Преобразует строку из набора символов, определяемого изготовителем оборудования, в ansi или строку расширенных символов. Предупреждение Не используйте. (Юникод)

[OffsetRect](#)

Функция OffsetRect перемещает указанный прямоугольник на указанные смещения.

[OpenClipboard](#)

Открывает буфер обмена для проверки и запрещает другим приложениям изменять содержимое буфера обмена.

[OpenDesktopA](#)

Открывает указанный объект рабочего стола. (ANSI)

[OpenDesktopW](#)

Открывает указанный объект рабочего стола. (Юникод)

[OpenIcon](#)

Восстанавливает свернутое (заковое) окно до предыдущего размера и положения; Затем он активирует окно.

[OpenInputDesktop](#)

Открывает рабочий стол, который получает данные, введенные пользователем.

[OpenWindowStationA](#)

Открывает указанную оконную станцию. (ANSI)

[OpenWindowStationW](#)

Открывает указанную станцию окон. (Юникод)

[PackTouchHitTestingProximityEvaluation](#)

Возвращает оценку близкого взаимодействия и скорректированные координаты сенсорной точки в виде упакованного значения для обратного вызова WM_TOUCHHITTESTING.

[PaintDesktop](#)

Функция PaintDesktop заполняет область обрезки в указанном контексте устройства шаблоном рабочего стола или фоном. Функция предоставляется в основном для рабочих столов оболочки.

[PeekMessageA](#)

Отправляет входящие сообщения без очереди, проверяет очередь сообщений потока на наличие отправленного сообщения и извлекает сообщение (если таковые существуют). (ANSI)

[PeekMessageW](#)

Отправляет входящие сообщения без очереди, проверяет очередь сообщений потока на наличие отправленного сообщения и извлекает сообщение (если таковые существуют). (Юникод)

[PhysicalToLogicalPoint](#)

Преобразует физические координаты точки в окне в логические.

[PhysicalToLogicalPointForPerMonitorDPI](#)

Преобразует точку в окне из физических координат в логические, независимо от уровня точек на дюйм (точек на дюйм) для вызывающего объекта.

[POINTSTOPOINT](#)

Макрос POINTSTOPOINT копирует содержимое структуры POINTS в структуру POINT.

[POINTTOPoints](#)

Макрос POINTTOPoints преобразует структуру POINT в структуру POINTS.

[PostMessageA](#)

Помещает (публикует) сообщение в очередь сообщений, связанную с потоком, создающим указанное окно, и возвращается без ожидания обработки сообщения потоком. (ANSI)

[PostMessageW](#)

Помещает (публикует) сообщение в очередь сообщений, связанную с потоком, создающим указанное окно, и возвращается без ожидания обработки сообщения потоком. (Юникод)

[PostQuitMessage](#)

Указывает системе, что поток сделал запрос на завершение (завершение работы). Обычно он используется в ответ на WM_DESTROY сообщение.

[PostThreadMessageA](#)

Отправляет сообщение в очередь сообщений указанного потока. Он возвращает, не дожидаясь обработки сообщения потоком. (ANSI)

[PostThreadMessageW](#)

Отправляет сообщение в очередь сообщений указанного потока. Он возвращает, не дожидаясь обработки сообщения потоком. (Юникод)

[PrintWindow](#)

Функция PrintWindow копирует визуальное окно в указанный контекст устройства (DC), обычно это принтер.

[PrivateExtractIconsA](#)

Создает массив дескрипторов для значков, извлеченных из указанного файла. (ANSI)

[PrivateExtractIconsW](#)

Создает массив дескрипторов для значков, извлеченных из указанного файла. (Юникод)

[PtInRect](#)

Функция PtInRect определяет, находится ли указанная точка в указанном прямоугольнике.

[QueryDisplayConfig](#)

Функция QueryDisplayConfig извлекает сведения обо всех возможных путях отображения для всех устройств отображения или представлений в текущем параметре.

[RealChildWindowFromPoint](#)

Извлекает дескриптор дочернего окна в указанной точке. Поиск ограничен непосредственными дочерними окнами; Внуки и более глубокие окна потомков не ищутся.

[RealGetWindowClassA](#)

Извлекает строку, указывающую тип окна. (ANSI)

[RealGetWindowClassW](#)

Извлекает строку, указывающую тип окна. (Юникод)

[RedrawWindow](#)

Функция RedrawWindow обновляет указанный прямоугольник или область в клиентской области окна.

[RegisterClassA](#)

Регистрирует класс окна для последующего использования в вызовах функции CreateWindow или CreateWindowEx. (RegisterClassA)

[RegisterClassExA](#)

Регистрирует класс окна для последующего использования в вызовах функции CreateWindow или CreateWindowEx. (RegisterClassExA)

[RegisterClassExW](#)

Регистрирует класс окна для последующего использования в вызовах функции CreateWindow или CreateWindowEx. (RegisterClassExW)

[RegisterClassW](#)

Регистрирует класс окна для последующего использования в вызовах функции CreateWindow или CreateWindowEx. (RegisterClassW)

[RegisterClipboardFormatA](#)

Регистрирует новый формат буфера обмена. Затем этот формат можно использовать в качестве допустимого формата буфера обмена. (ANSI)

[RegisterClipboardFormatW](#)

Регистрирует новый формат буфера обмена. Затем этот формат можно использовать в качестве допустимого формата буфера обмена. (Юникод)

[RegisterDeviceNotificationA](#)

Регистрирует устройство или тип устройства, для которого окно будет получать уведомления. (ANSI)

[RegisterDeviceNotificationW](#)

Регистрирует устройство или тип устройства, для которого окно будет получать уведомления. (Юникод)

[RegisterForTooltipDismissNotification](#)

Позволяет приложениям или платформам пользовательского интерфейса регистрировать и отменять регистрацию окон для получения уведомлений о закрытии окон всплывающих подсказок.

[RegisterHotKey](#)

Определяет горячий ключ на уровне системы.

[RegisterPointerDeviceNotifications](#)

Регистрирует окно для обработки уведомлений WM_POINTERDEVICECHANGE, WM_POINTERDEVICEINRANGE и WM_POINTERDEVICEOUTOFRANGE указателя.

[RegisterPointerInputTarget](#)

Позволяет вызывающей объекту зарегистрировать целевое окно, в которое перенаправляются все входные данные указателя указанного типа.

[RegisterPointerInputTargetEx](#)

Параметр RegisterPointerInputTargetEx может быть изменен или недоступен. Вместо этого используйте RegisterPointerInputTarget.

[RegisterPowerSettingNotification](#)

Регистрирует приложение для получения уведомлений о параметрах питания для конкретного события параметров питания.

[RegisterRawInputDevices](#)

Регистрирует устройства, которые предоставляют необработанные входные данные.

[RegisterShellHookWindow](#)

Регистрирует указанное окно оболочки для получения определенных сообщений о событиях или уведомлениях, которые полезны для приложений оболочки.

[RegisterSuspendResumeNotification](#)

Регистрируется для получения уведомлений о приостановке или возобновлении работы системы. Аналогично PowerRegisterSuspendResumeNotification, но работает в пользовательском режиме и может принимать дескриптор окна.

[RegisterTouchHitTestingWindow](#)

Регистрирует окно для обработки уведомления WM_TOUCHHITTESTING.

[RegisterTouchWindow](#)

Регистрирует окно с поддержкой сенсорного ввода.

[RegisterWindowMessageA](#)

Определяет новое сообщение окна, которое гарантированно будет уникальным в системе. Значение сообщения можно использовать при отправке или публикации сообщений. (ANSI)

[RegisterWindowMessageW](#)

Определяет новое сообщение окна, которое гарантированно будет уникальным в системе. Значение сообщения можно использовать при отправке или публикации сообщений. (Юникод)

[ReleaseCapture](#)

Освобождает захват мыши из окна в текущем потоке и восстанавливает нормальную обработку ввода с помощью мыши.

[ReleaseDC](#)

Функция ReleaseDC освобождает контекст устройства (DC), освобождая его для использования другими приложениями. Влияние функции ReleaseDC зависит от типа контроллера домена. Он освобождает только общие контроллеры домена и контроллеры домена окон. Он не влияет на класс или частные контроллеры домена.

[RemoveClipboardFormatListener](#)

Удаляет заданное окно из списка прослушивателя формата буфера обмена, поддерживаемого системой.

[RemoveMenu](#)

Удаляет элемент меню или отсоединяет подменю от указанного меню.

[RemovePropa](#)

Удаляет запись из списка свойств указанного окна. Указанная символьная строка определяет запись, которая должна быть удалена. (ANSI)

[RemovePropw](#)

Удаляет запись из списка свойств указанного окна. Указанная символьная строка определяет запись, которая должна быть удалена. (Юникод)

[ReplyMessage](#)

Отвечает на сообщение, отправленное из другого потока функцией SendMessage.

[ScreenToClient](#)

Функция ScreenToClient преобразует экранные координаты указанной точки на экране в координаты клиентской области.

[ScrollIDC](#)

Функция ScrollIDC прокручивает прямоугольник битов по горизонтали и вертикали.

[ScrollWindow](#)

Функция ScrollWindow прокручивает содержимое клиентской области указанного окна.

[ScrollWindowEx](#)

Функция ScrollWindowEx прокручивает содержимое клиентской области указанного окна.

[SendDlgItemMessageA](#)

Отправляет сообщение указанному элементу управления в диалоговом окне. (ANSI)

[SendDlgItemMessageW](#)

Отправляет сообщение указанному элементу управления в диалоговом окне. (Юникод)

[SendInput](#)

Синтезирует нажатия клавиш, движения мыши и нажатия кнопок.

[SendMessage](#)

Функция SendMessage отправляет указанное сообщение в окно или окна. (Функция SendMessage)

[SendMessageA](#)

Отправляет указанное сообщение в окно или окна. Функция SendMessage вызывает процедуру окна для указанного окна и не возвращается, пока эта процедура не обработает сообщение. (SendMessageA)

[SendMessageCallbackA](#)

Отправляет указанное сообщение в окно или окна. (SendMessageCallbackA)

[SendMessageCallbackW](#)

Отправляет указанное сообщение в окно или окна. (SendMessageCallbackW)

[SendMessageTimeoutA](#)

Отправляет указанное сообщение в одно или несколько окон. (ANSI)

[SendMessageTimeoutW](#)

Отправляет указанное сообщение в одно или несколько окон. (Юникод)

[SendMessageW](#)

Функция SendMessageW (Юникод) отправляет указанное сообщение в окно или окна. (SendMessageW)

[SendNotifyMessageA](#)

Отправляет указанное сообщение в окно или окна. (SendNotifyMessageA)

[SendNotifyMessageW](#)

Отправляет указанное сообщение в окно или окна. (SendNotifyMessageW)

[SetActiveWindow](#)

Активирует окно. Окно должно быть присоединено к очереди сообщений вызывающего потока.

[SetAdditionalForegroundBoostProcesses](#)

SetAdditionalForegroundBoostProcesses — это API помощи по повышению производительности, помогающий приложениям с многопроцессной моделью приложений, в которой несколько процессов участвуют в работе переднего плана как в виде данных, так и для отрисовки.

[SetCapture](#)

Задает для захвата мыши указанное окно, принадлежащее текущему потоку.

[SetCaretBlinkTime](#)

Задает для времени мигания курсора указанное число миллисекундах. Время мигания — это затраченное время (в миллисекундах), необходимое для инвертировать пиксели курсора.

[SetCaretPos](#)

Перемещает курсор в указанные координаты. Если окно, владеющее курсором, было создано в стиле класса CS_OWNDC, то указанные координаты подчиняются режиму сопоставления контекста устройства, связанного с этим окном.

[SetClassLongA](#)

Заменяет указанное 32-битовое значение (long) с указанным смещением в дополнительную память класса или структуру WNDCLASSEX для класса, которому принадлежит указанное окно. (ANSI)

[SetClassLongPtrA](#)

Заменяет указанное значение с указанным смещением в дополнительной памяти класса или структуре WNDCLASSEX для класса, которому принадлежит указанное окно. (ANSI)

[SetClassLongPtrW](#)

Заменяет указанное значение с указанным смещением в дополнительной памяти класса или структуре WNDCLASSEX для класса, которому принадлежит указанное окно. (Юникод)

[SetClassLongW](#)

Заменяет указанное 32-разрядное значение (long) с указанным смещением в дополнительную память класса или структуру WNDCLASSEX для класса, которому принадлежит указанное окно. (Юникод)

[SetClassWord](#)

Заменяет 16-битовое значение (WORD) с указанным смещением в дополнительную память класса для класса окна, которому принадлежит указанное окно.

[SetClipboardData](#)

Помещает данные в буфер обмена в указанном формате буфера обмена.

[SetClipboardViewer](#)

Добавляет указанное окно в цепочку средств просмотра буфера обмена. Окна средства просмотра буфера обмена получают WM_DRAWCLIPBOARD сообщение при каждом изменении содержимого буфера обмена. Эта функция используется для обратной совместимости с более ранними версиями Windows.

[SetCoalescableTimer](#)

Создает таймер с указанным значением времени ожидания и задержкой допуска объединения.

[SetCursor](#)

Задает фигуру курсора.

[SetCursorPos](#)

Перемещает курсор на указанные экранные координаты.

[SetDialogControlDpiChangeBehavior](#)

Переопределяет поведение масштабирования по умолчанию для каждого монитора DPI дочернего окна в диалоговом окне.

[SetDialogDpiChangeBehavior](#)

Диалоговые окна в контекстах Per-Monitor версии 2 автоматически масштабируются на дюйм. Этот метод позволяет настроить поведение изменения DPI.

[SetDisplayAutoRotationPreferences](#)

Задает параметры автоматического поворота экрана для текущего процесса.

[SetDisplayConfig](#)

Функция SetDisplayConfig изменяет топологию отображения, исходный и целевой режимы, включив исключительно указанные пути в текущем сеансе.

[SetDlgItemInt](#)

Задает текст элемента управления в диалоговом окне строковое представление указанного целочисленного значения.

[SetDlgItemTextA](#)

Задает заголовок или текст элемента управления в диалоговом окне. (ANSI)

[SetDlgItemTextW](#)

Задает заголовок или текст элемента управления в диалоговом окне. (Юникод)

[SetDoubleClickTime](#)

Задает время двойного щелчка мыши.

[SetFocus](#)

Устанавливает фокус клавиатуры в указанное окно. Окно должно быть подключено к очереди сообщений вызывающего потока.

[SetForegroundWindow](#)

Переносит поток, создавший указанное окно, на передний план и активирует окно.

[SetGestureConfig](#)

Настраивает сообщения, отправляемые из окна для жестов Windows Touch.

[SetKeyboardState](#)

Копирует массив состояний клавиш клавиатуры в таблицу входных данных клавиатуры вызывающего потока. Это та же таблица, к которым обращаются функции GetKeyboardState и GetKeyState. Изменения, внесенные в эту таблицу, не влияют на ввод с клавиатуры в любой другой поток.

[SetLastErrorEx](#)

Задает код последней ошибки.

[SetLayeredWindowAttributes](#)

Задает ключ цвета прозрачности многослойного окна.

[SetMenu](#)

Назначает новое меню указанному окну.

[SetMenuItemContextHelpId](#)

Связывает идентификатор контекста справки с меню.

[SetMenuItemDefault](#)

Задает пункт меню по умолчанию для указанного меню.

[SetMenuItemInfo](#)

Задает сведения для указанного меню.

[SetMenuItemBitmaps](#)

Связывает указанное растровое изображение с элементом меню. Независимо от того, выбран ли элемент меню или снят, система отображает соответствующее растровое изображение рядом с элементом меню.

[SetMenuItemInfoA](#)

Изменяет сведения об элементе меню. (ANSI)

[SetMenuItemInfoW](#)

Изменяет сведения об элементе меню. (Юникод)

[SetMessageExtraInfo](#)

Задает дополнительные сведения о сообщении для текущего потока.

Setparent	Изменяет родительское окно указанного дочернего окна.
SetPhysicalCursorPos	Задает положение курсора в физических координатах.
SetProcessDefaultLayout	Изменяет макет по умолчанию, если окна создаются без родителей или владельцев только для текущего запущенного процесса.
SetProcessDPIAware	Программа SetProcessDPIAware может быть изменена или недоступна. Вместо этого используйте SetProcessDPIAwareness.
SetProcessDpiAwarenessContext	Задает для текущего процесса заданный контекст осведомленности с точками на дюйм (dpi). Контексты осведомленности о DPI относятся к значению DPI_AWARENESS_CONTEXT.
SetProcessRestrictionExemption	Исключает вызывающий процесс из ограничений, препятствующих взаимодействию классических процессов со средой приложения Магазина Windows. Эта функция используется средствами разработки и отладки.
SetProcessWindowStation	Назначает вызывающему процессу указанную станцию окна.
SetPropA	Добавляет новую запись или изменяет существующую запись в списке свойств указанного окна. (ANSI)
SetPropW	Добавляет новую запись или изменяет существующую запись в списке свойств указанного окна. (Юникод)
SetRect	Функция SetRect задает координаты указанного прямоугольника. Это эквивалентно назначению левых, верхних, правых и нижних аргументов соответствующим членам структуры RECT.

[SetRectEmpty](#)

Функция SetRectEmpty создает пустой прямоугольник, в котором все координаты равны нулю.

[SetScrollInfo](#)

Функция SetScrollInfo задает параметры полосы прокрутки, включая минимальную и максимальную позиции прокрутки, размер страницы и положение поля прокрутки (большого пальца). Функция также перерисовывает полосу прокрутки при запросе.

[SetScrollPos](#)

Функция SetScrollPos задает положение поля прокрутки (большого пальца) на указанной полосе прокрутки и при необходимости перерисовывает полосу прокрутки, чтобы отразить новое положение поля прокрутки.

[SetScrollRange](#)

Функция SetScrollRange задает минимальное и максимальное положение поля прокрутки для указанной полосы прокрутки.

[SetSysColors](#)

Задает цвета для указанных элементов отображения.

[SetSystemCursor](#)

Позволяет приложению настраивать системные курсоры. Он заменяет содержимое системного курсора, указанного параметром id, содержимым курсора, заданного параметром hcur, а затем уничтожает hcur.

[SetThreadCursorCreationScaling](#)

Задает масштаб DPI, для которого предназначены курсоры, создаваемые в этом потоке. Это значение учитывается при масштабировании курсора для конкретного монитора, на котором он отображается.

[SetThreadDesktop](#)

Назначает указанный рабочий стол вызывающему потоку. Все последующие операции на рабочем столе используют права доступа, предоставленные рабочему столу.

[SetThreadDpiAwarenessContext](#)

Задайте для текущего потока сведения о DPI указанное значение.

[SetThreadDpiHostingBehavior](#)

Задает DPI_HOSTING_BEHAVIOR потока. Это позволяет окнам, созданным в потоке, размещать дочерние окна с другой DPI_AWARENESS_CONTEXT.

[SetTimer](#)

Создает таймер с указанным значением времени ожидания.

[SetUserObjectInformationA](#)

Задает сведения о указанной оконной станции или объекте desktop. (ANSI)

[SetUserObjectInformationW](#)

Задает сведения о указанной оконной станции или объекте desktop. (Юникод)

[SetUserObjectSecurity](#)

Задает безопасность объекта пользователя. Это может быть, например, окно или беседа DDE.

[SetWindowContextHelpId](#)

Связывает идентификатор контекста справки с указанным окном.

[SetWindowDisplayAffinity](#)

Сохраняет параметр сопоставления отображения в режиме ядра в hWnd, связанном с окном.

[SetWindowFeedbackSetting](#)

Задает конфигурацию обратной связи для окна.

[SetWindowLongA](#)

Изменяет атрибут указанного окна. Функция также задает 32-разрядное значение (long) с указанным смещением в дополнительную память окна. (ANSI)

[SetWindowLongPtrA](#)

Изменяет атрибут указанного окна. (ANSI)

[SetWindowLongPtrW](#)

Изменяет атрибут указанного окна. (Юникод)

[SetWindowLongW](#)

Изменяет атрибут указанного окна. Функция также задает 32-разрядное значение (long) с указанным смещением в дополнительную память окна. (Юникод)

[SetWindowPlacement](#)

Задает состояние отображения и восстановленные, свернутые и развернутые позиции указанного окна.

[SetWindowPos](#)

Изменяет размер, положение и порядок Z дочернего, всплывающего окна или окна верхнего уровня. Эти окна упорядочены в соответствии с их внешним видом на экране. Самое верхнее окно получает наивысший ранг и является первым окном в порядке Z.

[SetWindowRgn](#)

Функция SetWindowRgn задает область окна.

[SetWindowsHookExA](#)

Устанавливает определяемую приложением процедуру перехватчика в цепочку перехватчиков. (ANSI)

[SetWindowsHookExW](#)

Устанавливает определяемую приложением процедуру перехватчика в цепочку перехватчиков. (Юникод)

[SetWindowTextA](#)

Изменяет текст заголовка указанного окна (если он имеется). Если указанное окно является элементом управления, текст элемента управления изменяется. Однако SetWindowText не может изменить текст элемента управления в другом приложении. (ANSI)

[SetWindowTextW](#)

Изменяет текст заголовка указанного окна (если он имеется). Если указанное окно является элементом управления, текст элемента управления изменяется. Однако SetWindowText не может изменить текст элемента управления в другом приложении. (Юникод)

[SetWinEventHook](#)

Задает функцию перехватчика событий для диапазона событий.

ShowCaret

Делает курсор видимым на экране в текущей позиции курсора. Когда курсор становится видимым, он начинает мигать автоматически.

ShowCursor

Отображает или скрывает курсор. (ShowCursor)

ShowOwnedPopups

Показывает или скрывает все всплывающие окна, принадлежащие указанному окну.

ShowScrollBar

Функция ShowScrollBar отображает или скрывает указанную полосу прокрутки.

Showwindow

Задает состояние отображения указанного окна.

ShowWindowAsync

Задает состояние отображения окна, не дожидаясь завершения операции.

ShutdownBlockReasonCreate

Указывает, что система не может быть завершена, и задает строку причины, отображаемую для пользователя, если запущено завершение работы системы.

ShutdownBlockReasonDe shutdown

Указывает, что система может быть выключена, и освобождает строку причины.

ShutdownBlockReasonQuery

Извлекает строку причины, заданную функцией ShutdownBlockReasonCreate.

SkipPointerFrameMessages

Определяет, какой входной кадр указателя создал последнее полученное сообщение для указанного указателя, и удаляет все входные сообщения указателя в очереди (не восстановленные), созданные из того же входного кадра указателя.

SoundSentry

Активирует визуальный сигнал, указывающий, что воспроизводится звук.

[SubtractRect](#)

Функция SubtractRect определяет координаты прямоугольника, сформированного путем вычитания одного прямоугольника из другого.

[SwapMouseButton](#)

Изменяет или восстанавливает значение левой и правой кнопок мыши.

[SwitchDesktop](#)

Делает указанный рабочий стол видимым и активирует его. Это позволяет рабочему столу получать входные данные от пользователя.

[SwitchToThisWindow](#)

Переключает фокус на указанное окно и перемещает его на передний план.

[SystemParametersInfoA](#)

Получает или задает значение одного из системных параметров. (ANSI)

[SystemParametersInfoForDpi](#)

Извлекает значение одного из системных параметров с учетом указанного значения DPI.

[SystemParametersInfoW](#)

Получает или задает значение одного из системных параметров. (Юникод)

[TabbedTextOutA](#)

Функция TabbedTextOut записывает символьную строку в указанном расположении, разворачивая табуляции до значений, указанных в массиве позиций табуляции. Текст записывается выбранным шрифтом, цветом фона и цветом текста. (ANSI)

[TabbedTextOutW](#)

Функция TabbedTextOut записывает символьную строку в указанном расположении, разворачивая табуляции до значений, указанных в массиве позиций табуляции. Текст записывается выбранным шрифтом, цветом фона и цветом текста. (Юникод)

[TileWindows](#)

Плитки указанных дочерних окон указанного родительского окна.

[ToAscii](#)

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующий символ или символы.

[ToAsciiEx](#)

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующий символ или символы. Функция преобразует код с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода.

[TOUCH_COORD_TO_PIXEL](#)

Преобразует координаты касания в пиксели.

[ToUnicode](#)

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы Юникода. (ToUnicode)

[ToUnicodeEx](#)

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы Юникода. (ToUnicodeEx)

[TrackMouseEvent](#)

Публикует сообщения, когда указатель мыши покидает окно или наносит указатель мыши на окно в течение указанного периода времени.

[TrackPopupMenu](#)

Отображает контекстное меню в указанном расположении и отслеживает выбор элементов в меню. Контекстное меню может отображаться в любом месте экрана.

[TrackPopupMenuEx](#)

Отображает контекстное меню в указанном расположении и отслеживает выбор элементов в контекстном меню. Контекстное меню может отображаться в любом месте экрана.

[TranslateAcceleratorA](#)

Обрабатывает клавиши ускорителя для команд меню. (ANSI)

[TranslateAcceleratorW](#)

Обрабатывает клавиши ускорителя для команд меню. (Юникод)

[TranslateMDISysAccel](#)

Обрабатывает нажатия клавиш акселератора для команд меню окна дочерних окон многодокументного интерфейса (MDI), связанных с указанным окном клиента MDI.

[TranslateMessage](#)

Преобразует сообщения с виртуальным ключом в символьные сообщения. Символьные сообщения отправляются в очередь сообщений вызывающего потока, чтобы прочитать их в следующий раз, когда поток вызывает функцию GetMessage или PeekMessage.

[UnhookWindowsHookEx](#)

Удаляет процедуру перехватчика, установленную в цепочке перехватчиков функцией SetWindowsHookEx.

[UnhookWinEvent](#)

Удаляет функцию перехватчика событий, созданную при предыдущем вызове SetWinEventHook.

[UnionRect](#)

Функция UnionRect создает объединение двух прямоугольников. Объединение — это наименьший прямоугольник, содержащий оба исходных прямоугольника.

[UnloadKeyboardLayout](#)

Выгружает идентификатор входного языкового стандарта (прежнее название — раскладка клавиатуры).

[Отмена регистрацииClassA](#)

Отменяет регистрацию класса окна, освобождая память, необходимую для класса . (ANSI)

[Отмена регистрацииClassW](#)

Отменяет регистрацию класса окна, освобождая память, необходимую для класса . (Юникод)

[UnregisterDeviceNotification](#)

Закрывает указанный дескриптор уведомления устройства.

[Отмена регистрацииHotKey](#)

Освобождает горячий ключ, ранее зарегистрированный вызывающим потоком.

[Отмена регистрацииPointerInputTarget](#)

Позволяет вызывающей объекту отменить регистрацию целевого окна, в которое перенаправляются все входные данные указателя указанного типа.

[Отмена регистрацииPointerInputTargetEx](#)

UnregisterPointerInputTargetEx может быть изменен или недоступен. Вместо этого используйте UnregisterPointerInputTarget.

[Отмена регистрацииPowerSettingNotification](#)

Отменяет регистрацию уведомления о настройке питания.

[Отмена регистрацииSuspendResumeNotification](#)

Отменяет регистрацию для получения уведомления о приостановке или возобновлении работы системы. Аналогично PowerUnregisterSuspendResumeNotification, но работает в пользовательском режиме.

[Отменить регистрациюTouchWindow](#)

Регистрирует окно как не поддерживающее сенсорный режим.

[UpdateLayeredWindow](#)

Обновляет положение, размер, форму, содержимое и прозрачность многослойного окна.

[UpdateWindow](#)

Функция UpdateWindow обновляет клиентскую область указанного окна, отправляя WM_PAINT сообщение в окно, если область обновления окна не пуста.

[UserHandleGrantAccess](#)

Предоставляет или запрещает доступ к дескриптору объекта User заданию с ограничением пользовательского интерфейса.

[ValidateRect](#)

Функция ValidateRect проверяет клиентную область в прямоугольнике, удалив прямоугольник из области обновления указанного окна.

[ValidateRgn](#)

Функция ValidateRgn проверяет клиентную область в регионе, удаляя регион из текущей области обновления указанного окна.

[VkKeyScanA](#)

Преобразует символ в соответствующий код виртуальной клавиши и состояние сдвига для текущей клавиатуры. (ANSI)

[VkKeyScanExA](#)

Преобразует символ в соответствующий код виртуального ключа и состояние сдвига. Функция переводит символ с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода. (ANSI)

[VkKeyScanExW](#)

Преобразует символ в соответствующий код виртуального ключа и состояние сдвига. Функция переводит символ с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода. (Юникод)

[VkKeyScanW](#)

Преобразует символ в соответствующий код виртуальной клавиши и состояние сдвига для текущей клавиатуры. (Юникод)

[WaitForInputIdle](#)

Ожидает, пока указанный процесс завершит обработку своих начальных входных данных, ожидает ввода данных пользователем без ожидающих ввода данных или до истечения интервала ожидания.

[WaitMessage](#)

Возвращает управление другим потокам, если поток не содержит других сообщений в очереди сообщений. Функция WaitMessage приостанавливает поток и не возвращает его, пока новое сообщение не будет помещено в очередь сообщений потока.

[WindowFromDC](#)

Функция WindowFromDC возвращает дескриптор для окна, связанного с указанным контекстом устройства отображения (DC). Функции вывода, использующие указанный контекст устройства, зарисовываются в этом окне.

[WindowFromPhysicalPoint](#)

Извлекает дескриптор окна, содержащего указанную физическую точку.

[WindowFromPoint](#)

Извлекает дескриптор для окна, содержащего указанную точку.

[WinHelpA](#)

Запускает справку Windows (Winhelp.exe) и передает дополнительные данные, указывающие на характер справки, запрашиваемой приложением. (ANSI)

[WinHelpW](#)

Запускает справку Windows (Winhelp.exe) и передает дополнительные данные, указывающие на характер справки, запрашиваемой приложением. (Юникод)

[wsprintfA](#)

Записывает отформатированные данные в указанный буфер. (ANSI)

[wsprintfW](#)

Записывает отформатированные данные в указанный буфер. (Юникод)

[vwsprintfA](#)

Записывает форматированные данные в указанный буфер, используя указатель на список аргументов. (ANSI)

[vwsprintfW](#)

Записывает форматированные данные в указанный буфер, используя указатель на список аргументов. (Юникод)

Функции обратного вызова

[DLGPROC](#)

Определяемая приложением функция обратного вызова, используемая с семействами функций CreateDialog и DialogBox.

[DRAWSTATEPROC](#)

Функция DrawStateProc — это определяемая приложением функция обратного вызова, которая отрисовывает сложное изображение для функции DrawState.

[EDITWORDBREAKPROCA](#)

Определяемая приложением функция обратного вызова, используемая с сообщением EM_SETWORDBREAKPROC. (ANSI)

[EDITWORDBREAKPROCW](#)

Определяемая приложением функция обратного вызова, используемая с сообщением EM_SETWORDBREAKPROC. (Юникод)

[GRAYSTRINGPROC](#)

Функция OutputProc — это определяемая приложением функция обратного вызова, используемая вместе с функцией GrayString.

[HOOKPROC](#)

Определяемая приложением или библиотекой функция обратного вызова, используемая с функцией SetWindowsHookEx. Система вызывает эту функцию после вызова функции SendMessage. Процедура перехватчика может проверить сообщение; он не может изменить его.

[MONITORENUMPROC](#)

Функция MonitorEnumProc — это определяемая приложением функция обратного вызова, вызываемая функцией EnumDisplayMonitors.

[MSGBOXCALLBACK](#)

Функция обратного вызова, определяемая в приложении, которая обрабатывает события справки для окна сообщения.

[PROOPENUMPROCA](#)

Определяемая приложением функция обратного вызова, используемая с функцией EnumProps. (ANSI)

[PROOPENUMPROCEXA](#)

Определяемая приложением функция обратного вызова, используемая с функцией EnumPropsEx. (ANSI)

[PROOPENUMPROCEWX](#)

Определяемая приложением функция обратного вызова, используемая с функцией EnumPropsEx. (Юникод)

[PROOPENUMPROCW](#)

Определяемая приложением функция обратного вызова, используемая с функцией EnumProps. (Юникод)

SENDASYNCPROC

Определяемая приложением функция обратного вызова, используемая с функцией SendMessageCallback.

TIMERPROC

Определяемая приложением функция обратного вызова, которая обрабатывает WM_TIMER сообщения. Тип TIMERPROC определяет указатель на эту функцию обратного вызова. TimerProc — это заполнитель для имени функции, определяемой приложением.

WINEVENTPROC

Определяемая приложением функция обратного вызова (или перехватчика), которую система вызывает в ответ на события, созданные доступным объектом.

WNDPROC

Функция обратного вызова, определяемая в приложении, которая обрабатывает сообщения, отправленные в окно.

Структуры

ACCEL

Определяет ключ ускорителя, используемый в таблице ускорителей.

ACCESSTIMEOUT

Содержит сведения о периоде ожидания, связанном с функциями специальных возможностей.

ALTTABINFO

Содержит сведения о состоянии окна переключения приложений (ALT+TAB).

ANIMATIONINFO

Описывает эффекты анимации, связанные с действиями пользователя.

AUDIODESCRIPTION

Содержит сведения, связанные с описаниями звука. Эта структура используется с функцией SystemParametersInfo, когда указано значение действия SPI_GETAUDIODESCRIPTION или SPI_SETAUDIODESCRIPTION.

BSMINFO

Содержит сведения об окне, которое отклонило запрос от BroadcastSystemMessageEx.

CBT_CREATEWNDA

Содержит сведения, передаваемые WH_CBT процедуре перехватчика CBTProc перед созданием окна. (ANSI)

CBT_CREATEWNDW

Содержит сведения, передаваемые WH_CBT процедуре перехватчика CBTProc перед созданием окна. (Юникод)

CBTACTIVATESTRUCT

Содержит сведения, передаваемые WH_CBT процедуре перехватчика CBTProc перед активацией окна.

CHANGEFILTERSTRUCT

Содержит расширенные сведения о результатах, полученные при вызове функции ChangeWindowMessageFilterEx.

CLIENTCREATESTRUCT

Содержит сведения о меню и первом дочернем окне многодокументного интерфейса (MDI) клиентского окна MDI.

COMBOBOXINFO

Содержит сведения о состоянии поля со списком.

COMPAREITEMSTRUCT

Предоставляет идентификаторы и данные, предоставленные приложением, для двух элементов в отсортированных, нарисованных владельцем списка или поля со списком.

COPYDATASTRUCT

Содержит данные, передаваемые другому приложению с помощью сообщения WM_COPYDATA.

[CREATESTRUCTA](#)

Определяет параметры инициализации, передаваемые в оконную процедуру приложения. Эти элементы идентичны параметрам функции CreateWindowEx. (ANSI)

[CREATESTRUCTW](#)

Определяет параметры инициализации, передаваемые в оконную процедуру приложения. Эти элементы идентичны параметрам функции CreateWindowEx. (Юникод)

[CURSORINFO](#)

Содержит глобальные сведения о курсоре.

[CURSORSHAPE](#)

Содержит сведения о курсоре.

[CWPRETSSTRUCT](#)

Определяет параметры сообщения, передаваемые в процедуру перехватчика WH_CALLWNDPROCRET CallWndRetProc.

[CWPSTRUCT](#)

Определяет параметры сообщения, передаваемые WH_CALLWNDPROC процедуре перехватчика CallWndProc.

[DEBUGHOOKINFO](#)

Содержит сведения об отладке, передаваемые WH_DEBUG процедуре перехватчика DebugProc.

[DELETEITEMSTRUCT](#)

Описывает удаленный список или элемент поля со списком.

[DLGITEMTEMPLATE](#)

Определяет размеры и стиль элемента управления в диалоговом окне. Одна или несколько из этих структур объединяются со структурой DLGTEMPLATE, чтобы сформировать стандартный шаблон для диалогового окна.

[DLGTEMPLATE](#)

Определяет размеры и стиль диалогового окна.

DRAWITEMSTRUCT

Предоставляет сведения, которые окно владельца использует для определения способа рисования нарисованного владельцем элемента управления или пункта меню.

DRAWTEXTPARAMS

Структура DRAWTEXTPARAMS содержит расширенные параметры форматирования для функции DrawTextEx.

EVENTMSG

Содержит сведения об аппаратном сообщении, отправляемом в очередь системных сообщений. Эта структура используется для хранения сведений о сообщениях функции обратного вызова JournalPlaybackProc.

FILTERKEYS

Содержит сведения о функции специальных возможностей FilterKeys, которая позволяет пользователю с ограниченными возможностями задать частоту повторения клавиатуры (RepeatKeys), задержку принятия (SlowKeys) и частоту отказов (BounceKeys).

FLASHWINFO

Содержит состояние вспышки для окна и количество раз, когда система должна мигать окно.

GESTURECONFIG

Возвращает и задает конфигурацию для включения сообщений жестов и тип этой конфигурации.

ЖЕСТОВАЯ ИНФОРМАЦИЯ

Хранит сведения о жесте.

GESTURENOTIFYSTRUCT

При передаче с помощью WM_GESTURENOTIFY сообщений передает сведения о жесте.

GUITHREADINFO

Содержит сведения о потоке графического пользовательского интерфейса.

HARDWAREINPUT

Содержит сведения о имитированном сообщении, созданном устройством ввода, которое отличается от клавиатуры или мыши.

[HELPINFO](#)

Содержит сведения об элементе, для которого была запрошена контекстная справка.

[HELPWININFOA](#)

Содержит размер и положение основного или дополнительного окна справки. Приложение может задать эти сведения, вызвав функцию WinHelp со значением HELP_SETWINPOS. (ANSI)

[HELPWININFOW](#)

Содержит размер и положение основного или дополнительного окна справки. Приложение может задать эти сведения, вызвав функцию WinHelp со значением HELP_SETWINPOS. (Юникод)

[ХАЙКОНТРАСТА](#)

Содержит сведения о функции специальных возможностей с высокой контрастностью. (ANSI)

[HIGHCONTRASTW](#)

Содержит сведения о функции специальных возможностей с высокой контрастностью. (Юникод)

[ICONINFO](#)

Содержит сведения о значке или курсоре.

[ICONINFOEXA](#)

Содержит сведения о значке или курсоре. Расширяет ICONINFO. Используется Методом GetIconInfoEx. (ANSI)

[ICONINFOEXW](#)

Содержит сведения о значке или курсоре. Расширяет ICONINFO. Используется Методом GetIconInfoEx. (Юникод)

[ICONMETRICS](#)

Содержит масштабируемые метрики, связанные со значками. Эта структура используется с функцией SystemParametersInfo при указании действия SPI_GETICONMETRICS или SPI_SETICONMETRICS. (ANSI)

[ICONMETRICSW](#)

Содержит масштабируемые метрики, связанные со значками. Эта структура используется с функцией `SystemParametersInfo` при указании действия `SPI_GETICONMETRICS` или `SPI_SETICONMETRICS`. (Юникод)

[INPUT](#)

Используется `SendInput` для хранения информации для синтеза событий ввода, таких как нажатия клавиш, перемещение мыши и щелчки мышью.

[INPUT_INJECTION_VALUE](#)

Содержит сведения о входных данных о внедрении.

[INPUT_MESSAGE_SOURCE](#)

Содержит сведения об источнике входного сообщения.

[INPUT_TRANSFORM](#)

Определяет матрицу, представляющую преобразование для потребителя сообщения.

[KBDLLHOOKSTRUCT](#)

Содержит сведения о низкоуровневом событии ввода с помощью клавиатуры.

[KEYBDINPUT](#)

Содержит сведения о событии имитации клавиатуры.

[LASTINPUTINFO](#)

Содержит время последнего ввода.

[MDICREATESTRUCTA](#)

Содержит сведения о классе, заголовке, владельце, расположении и размере дочернего окна MDI. (ANSI)

[MDICREATESTRUCTW](#)

Содержит сведения о классе, заголовке, владельце, расположении и размере дочернего окна MDI. (Юникод)

[MDINEXTMENU](#)

Содержит сведения о меню для активации.

MEASUREITEMSTRUCT

Информирует систему о размерах элемента управления или пункта меню, нарисованного владельцем. Это позволяет системе правильно обрабатывать взаимодействие пользователя с элементом управления.

MENUBARINFO

Содержит сведения о строке меню.

MENUGETOBJECTINFO

Содержит сведения о меню, в котором находится курсор мыши.

MENUINFO

Содержит сведения о меню.

MENUITEMINFOA

Содержит сведения об элементе меню. (MENUITEMINFOA)

MENUITEMINFOW

Содержит сведения об элементе меню. (MENUITEMINFOW)

MENUITEMTEMPLATE

Определяет пункт меню в шаблоне меню.

MENUITEMTEMPLATEHEADER

Определяет заголовок для шаблона меню. Полный шаблон меню состоит из заголовка и одного или нескольких списков пунктов меню.

СВЕРНУТЫЕ МЕТРИКИ

Содержит масштабируемые метрики, связанные с свернутыми окнами.

MINMAXINFO

Содержит сведения о максимальном размере и положении окна, а также его минимальном и максимальном размерах отслеживания.

MONITORINFO

Структура MONITORINFO содержит сведения о мониторе дисплея. Функция GetMonitorInfo хранит сведения в структуре MONITORINFO или MONITORINFOEX. Структура MONITORINFO — это подмножество структуры MONITORINFOEX.

[MONITORINFOEXA](#)

Структура MONITORINFOEX содержит сведения о мониторе дисплея. Функция GetMonitorInfo сохраняет информацию в структуре MONITORINFOEX или MONITORINFO. Структура MONITORINFOEX — это надмножество структуры MONITORINFO. (ANSI)

[MONITORINFOEXW](#)

Структура MONITORINFOEX содержит сведения о мониторе дисплея. Функция GetMonitorInfo сохраняет информацию в структуре MONITORINFOEX или MONITORINFO. Структура MONITORINFOEX — это надмножество структуры MONITORINFO. (Юникод)

[MOUSEHOOKSTRUCT](#)

Содержит сведения о событии мыши, переданном в процедуру перехватчика WH_MOUSE MouseProc.

[MOUSEHOOKSTRUCTEX](#)

Содержит сведения о событии мыши, переданном в процедуру перехватчика WH_MOUSE MouseProc. Это расширение структуры MOUSEHOOKSTRUCT, включающее сведения о перемещении колесика или использовании кнопки X.

[MOUSEINPUT](#)

Содержит сведения о имитированном событии мыши.

[КЛАВИШИ МЫШИ](#)

Содержит сведения о специальных возможностях MouseKeys.

[MOUSEMOVEPOINT](#)

Содержит сведения о расположении мыши в координатах экрана.

[MSG](#)

Содержит информацию сообщения из очереди сообщений потока.

[MSGBOXPARAMSA](#)

Содержит сведения, используемые для отображения окна сообщения. Эта структура используется в функции MessageBoxIndirect. (ANSI)

[MSGBOXPARAMSW](#)

Содержит сведения, используемые для отображения окна сообщения. Эта структура используется в функции MessageBoxIndirect. (Юникод)

[MSLHOOKSTRUCT](#)

Содержит сведения о низкоуровневом событии ввода мыши.

[MULTIKEYHELPA](#)

Указывает ключевое слово для поиска и таблицу ключевое слово, которую будет искать справка Windows. (ANSI)

[MULTIKEYHELPW](#)

Указывает ключевое слово для поиска и таблицу ключевое слово, которую будет искать справка Windows. (Юникод)

[NCCALCSIZE_PARAMS](#)

Содержит сведения, которые приложение может использовать при обработке сообщения WM_NCCALCSIZE для вычисления размера, положения и допустимого содержимого клиентской области окна.

[NMHDR](#)

Структура NMHDR содержит сведения о сообщении уведомления. (структура NMHDR)

[NONCLIENTMETRICSA](#)

Содержит масштабируемые метрики, связанные с неклиентской областью неминимизированного окна. (ANSI)

[NONCLIENTMETRICSW](#)

Содержит масштабируемые метрики, связанные с неклиентской областью неминимизированного окна. (Юникод)

[PAINTSTRUCT](#)

Структура PAINTSTRUCT содержит сведения для приложения. Эти сведения можно использовать для рисования клиентской области окна, принадлежащей этому приложению.

[POINTER_DEVICE_CURSOR_INFO](#)

Содержит сопоставления идентификаторов курсоров для устройств-указателей.

[POINTER_DEVICE_INFO](#)

Содержит сведения об устройстве указателя. Массив этих структур возвращается из функции GetPointerDevices. Одна структура возвращается из вызова функции GetPointerDevice.

[POINTER_DEVICE_PROPERTY](#)

Содержит свойства устройства на основе указателя (глобальные элементы устройства HID, которые соответствуют использованию HID).

[POINTER_INFO](#)

Содержит основные сведения о указателе, общие для всех типов указателей. Приложения могут получать эти сведения с помощью функций GetPointerInfo, GetPointerFrameInfo, GetPointerInfoHistory и GetPointerFrameInfoHistory.

[POINTER_PEN_INFO](#)

Определяет основные сведения о перо, общие для всех типов указателей.

[POINTER_TOUCH_INFO](#)

Определяет основные сенсорные сведения, общие для всех типов указателей.

[POINTER_TYPE_INFO](#)

Содержит сведения о типе входных данных указателя.

[POWERBROADCAST_SETTING](#)

Отправляется с событием настройки питания и содержит данные о конкретном изменении.

[RAWHID](#)

Описывает формат необработанных входных данных с устройства HID.

[RAWINPUT](#)

Содержит необработанные входные данные с устройства.

[RAWINPUTDEVICE](#)

Определяет сведения для необработанных устройств ввода.

[RAWINPUTDEVICELIST](#)

Содержит сведения о необработанном устройстве ввода.

[RAWINPUTHEADER](#)

Содержит сведения о заголовке, которые являются частью необработанных входных данных.

[RAWKEYBOARD](#)

Содержит сведения о состоянии клавиатуры.

[RAWMOUSE](#)

Содержит сведения о состоянии мыши.

[RID_DEVICE_INFO](#)

Определяет необработанные входные данные, поступающие с любого устройства.

[RID_DEVICE_INFO_HID](#)

Определяет необработанные входные данные, поступающие от указанного устройства HID.

[RID_DEVICE_INFO_KEYBOARD](#)

Определяет необработанные входные данные, поступающие с указанной клавиатурой.

[RID_DEVICE_INFO_MOUSE](#)

Определяет необработанные входные данные, поступающие от указанной мыши.

[SCROLLBARINFO](#)

Структура SCROLLBARINFO содержит сведения о полосе прокрутки.

[SCROLLINFO](#)

Структура SCROLLINFO содержит параметры полосы прокрутки, которые задаются функцией SetScrollInfo (или SBM_SETSCROLLINFO сообщением) или извлекаются функцией GetScrollInfo (или SBM_GETSCROLLINFO сообщением).

[SERIALKEYSA](#)

Содержит сведения о функции специальных возможностей SerialKeys, которая интерпретирует данные из средства связи, подключенного к последовательному порту, как команды, в результате чего система имитирует ввод с помощью клавиатуры и мыши. (ANSI)

[SERIALKEYSW](#)

Содержит сведения о функции специальных возможностей SerialKeys, которая интерпретирует данные из средства связи, подключенного к последовательному порту, как команды, в результате чего система имитирует ввод с помощью клавиатуры и мыши. (Юникод)

SOUNDSENTRYA

Содержит сведения о функции специальных возможностей SoundSentry. Если функция SoundSentry включена, компьютер отображает визуальное указание только при создании звука. (ANSI)

SOUNDSENTRYW

Содержит сведения о функции специальных возможностей SoundSentry. Если функция SoundSentry включена, компьютер отображает визуальное указание только при создании звука. (Юникод)

STICKYKEYS

Содержит сведения о функции специальных возможностей StickyKeys.

STYLESTRUCT

Содержит стили для окна.

TITLEBARINFO

Содержит сведения о строке заголовка.

TITLEBARINFOEX

Расширяет сведения, описанные в структуре TITLEBARINFO, включив координаты каждого элемента в строке заголовка.

КЛАВИШИ TOGGLEKEY

Содержит сведения о функции специальных возможностей ToggleKeys.

TOUCH_HIT_TESTING_INPUT

Содержит сведения об области сенсорного контакта, сообщаемой дигитайзером сенсорного ввода.

TOUCH_HIT_TESTING_PROXIMITY_EVALUATION

Содержит оценку проверки попадания, показывающую, является ли объект вероятной целью области касания относительно других объектов, пересекающих контактную область касания.

TOUCHINPUT

Инкапсулирует данные для сенсорного ввода.

TOUCHPREDICTIONPARAMETERS

Содержит сведения об аппаратном вводе, которые можно использовать для прогнозирования целевых объектов касания и компенсации аппаратной задержки при обработке сенсорного ввода и ввода жестов, содержащих данные о расстоянии и скорости.

TPMPARAMS

Содержит расширенные параметры для функции TrackPopupMenuEx.

TRACKMOUSEEVENT

Используется функцией TrackMouseEvent для отслеживания того, когда указатель мыши покидает окно или наводит указатель мыши на окно в течение указанного периода времени.

UPDATELAYEREDWINDOWINFO

Используется UpdateLayeredWindowIndirect для предоставления сведений о положении, размере, форме, содержимом и полупрозрачном окне.

USAGE_PROPERTIES

Содержит свойства устройства (глобальные элементы устройства HID, соответствующие использованию HID) для любого типа устройства ввода HID.

USEROBJECTFLAGS

Содержит сведения о оконной станции или дескрипторе рабочего стола.

WINDOWINFO

Содержит сведения о окне.

WINDOWPLACEMENT

Содержит сведения о размещении окна на экране.

WINDOWPOS

Содержит сведения о размере и положении окна.

WNDCLASSA

Содержит атрибуты класса окна, зарегистрированные функцией RegisterClass. (ANSI)

WNDCLASSEX A

Содержит сведения о классе окна. (ANSI)

WNDCLASSEXW

Содержит сведения о классе окна. (Юникод)

WNDCLASSW

Содержит атрибуты класса окна, зарегистрированные функцией RegisterClass. (Юникод)

WTSSESSION_NOTIFICATION

Предоставляет сведения об уведомлении об изменении сеанса. Служба получает эту структуру в функции HandlerEx в ответ на событие изменения сеанса.

Перечисления

AR_STATE

Указывает состояние автоматического поворота экрана для системы. Например, поддерживается ли автоматическая смена и включена ли она пользователем.

DIALOG_CONTROL_DPI_CHANGE_BEHAVIORS

Описывает поведение масштабирования DPI для каждого монитора для дочерних окон в диалоговых окнах. Значения в этом перечислении являются битовые поля и могут быть объединены.

DIALOG_DPI_CHANGE_BEHAVIORS

В контекстах "На монитор" версии 2 диалоговые окна будут автоматически реагировать на изменения DPI путем изменения размера и повторного вычисления положения дочерних окон (здесь называется изменением макета).

FEEDBACK_TYPE

Указывает визуальную обратную связь, связанную с событием.

INPUT_MESSAGE_DEVICE_TYPE

Тип устройства, отправляющего входное сообщение.

INPUT_MESSAGE_ORIGIN_ID

Идентификатор источника входного сообщения.

[ORIENTATION_PREFERENCE](#)

Указывает предпочтительный вариант ориентации экрана для процесса классического приложения.

[POINTER_BUTTON_CHANGE_TYPE](#)

Определяет изменение состояния кнопки, связанной с указателем.

[POINTER_DEVICE_CURSOR_TYPE](#)

Определяет типы курсоров устройства-указателя.

[POINTER_DEVICE_TYPE](#)

Определяет типы устройств указателя.

[POINTER_FEEDBACK_MODE](#)

Определяет поведение визуальной обратной связи, доступное для CreateSyntheticPointerDevice.

[tagPOINTER_INPUT_TYPE](#)

Определяет типы входных данных указателя.

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция ActivateKeyboardLayout (winuser.h)

Статья 27.08.2023

Задает идентификатор входного языкового стандарта (прежнее название — дескриптор раскладки клавиатуры) для вызывающего потока или текущего процесса. Идентификатор входного языкового стандарта указывает языковой стандарт, а также физический макет клавиатуры.

Синтаксис

C++

```
HKL ActivateKeyboardLayout(
    [in] HKL hkl,
    [in] UINT Flags
);
```

Параметры

[in] hkl

Тип: HKL

Входной идентификатор языкового стандарта для активации.

Идентификатор входного языкового стандарта должен быть загружен предыдущим вызовом функции [LoadKeyboardLayout](#). Этот параметр должен быть дескриптором раскладки клавиатуры или одним из следующих значений.

Значение	Значение
HKL_NEXT 1	Выбирает следующий идентификатор языкового стандарта в циклический список загруженных идентификаторов языкового стандарта, поддерживаемых системой.
HKL_PREV 0	Выбирает предыдущий идентификатор языкового стандарта из циклического списка загруженных идентификаторов языкового стандарта, поддерживаемых системой.

[in] Flags

Тип: **UINT**

Указывает способ активации идентификатора входного языкового стандарта. Этот параметр может принимать одно из указанных ниже значений.

Значение	Значение
KLF_REORDER 0x00000008	Если этот бит задан, циклический список системных идентификаторов загруженных языковых стандартов переупорядочен путем перемещения идентификатора языкового стандарта в начало списка. Если этот бит не задан, список поворачивается без изменения порядка. Например, если у пользователя активный идентификатор языкового стандарта на английском языке, а также загружены идентификаторы языкового стандарта на французском, немецком и испанском языках (в таком порядке), то активация идентификатора немецкого языкового стандарта с помощью набора битов KLF_REORDER приведет к следующему порядку: немецкий, английский, французский, испанский. Активация идентификатора немецкого языкового стандарта без KLF_REORDER битового набора приведет к следующему порядку: немецкий, испанский, английский, французский. Если загружено менее трех идентификаторов языкового стандарта, значение этого флага не имеет значения.
KLF_RESET 0x40000000	Если задано , но KLF_SHIFTLOCK не задано, состояние CAPS LOCK отключается путем повторного нажатия клавиши CAPS LOCK. Если также задано значение set и KLF_SHIFTLOCK , состояние CAPS LOCK отключается нажатием любой из клавиш SHIFT. Эти два метода являются взаимоисключающими, и параметр сохраняется как часть профиля пользователя в реестре.
KLF_SETFORPROCESS 0x00000100	Активирует указанный идентификатор языкового стандарта для всего процесса и отправляет сообщение WM_INPUTLANGCHANGE в фокус или активное окно текущего потока.
KLF_SHIFTLOCK 0x00010000	Используется с KLF_RESET . Пояснения см . в разделе KLF_RESET .

KLF_UNLOADPREVIOUS

Этот флаг не поддерживается. Вместо этого используйте функцию [UnloadKeyboardLayout](#).

Возвращаемое значение

Тип: **HKL**

Возвращаемое значение имеет тип **HKL**. Если функция выполнена успешно, возвращаемое значение является предыдущим идентификатором входного языкового стандарта. В противном случае значение равно нулю.

Чтобы получить расширенные сведения об ошибке, используйте функцию [GetLastError](#).

Комментарии

Эта функция влияет только на макет текущего процесса или потока.

Эта функция не ограничивается раскладками клавиатуры. Параметр *hkl* фактически является идентификатором входного языкового стандарта. Это более широкое понятие, чем раскладка клавиатуры, так как она также может охватывать преобразователь преобразования речи в текст, редактор метода ввода (IME) или любую другую форму ввода. Несколько входных идентификаторов языкового стандарта можно загрузить в любой момент времени, но раз активен только один. Загрузка нескольких идентификаторов входных языковых стандартов позволяет быстро переключаться между ними.

Если для каждого языкового стандарта разрешено несколько IME, передача входного идентификатора языкового стандарта, в котором высокое слово (дескриптор устройства) равно нулю, активирует первый IME в списке, относящемся к языковому стандарту.

Флаги **KLF_RESET** и **KLF_SHIFTLOCK** изменяют метод, с помощью которого отключается состояние CAPS LOCK. По умолчанию состояние CAPS LOCK отключается путем повторного нажатия клавиши CAPS LOCK. Если задано только **KLF_RESET**, состояние по умолчанию будет восстановлено. Если заданы **KLF_RESET** и **KLF_SHIFTLOCK**, состояние CAPS LOCK отключается нажатием клавиши CAPS LOCK. Эта функция используется для соответствия локальным стандартам поведения клавиатуры, а также для личных предпочтений.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

Основные понятия

[GetKeyboardLayoutName](#)

[Ввод с клавиатуры](#)

[LoadKeyboardLayout](#)

Справочные материалы

[UnloadKeyboardLayout](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция BlockInput (winuser.h)

Статья 27.08.2023

Блокирует доступ к приложениям событий ввода с помощью клавиатуры и мыши.

Синтаксис

C++

```
BOOL BlockInput(  
    [in] BOOL fBlockIt  
);
```

Параметры

[in] fBlockIt

Тип: **BOOL**

Назначение функции. Если этот параметр имеет значение **TRUE**, события ввода с помощью клавиатуры и мыши блокируются. Если этот параметр имеет значение **FALSE**, события клавиатуры и мыши разблокируются. Обратите внимание, что только поток, который заблокировал входные данные, может успешно разблокировать входные данные.

Возвращаемое значение

Тип: **BOOL**

Если функция выполняется успешно, возвращается ненулевое значение.

Если входные данные уже заблокированы, возвращаемое значение равно нулю. Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Если вход заблокирован, реальный физический ввод с мыши или клавиатуры не повлияет на синхронное состояние клавиши очереди ввода ([сообщается GetKeyState](#) и [GetKeyboardState](#)), а также не повлияет на состояние асинхронного ключа ([сообщает GetAsyncKeyState](#)). Однако поток, блокирующий ввод, может

повлиять на оба этих состояния ключей путем вызова [SendInput](#). Ни один другой поток не может это сделать.

Система разблокировать ввод будет в следующих случаях:

- Поток, который заблокировал ввод, неожиданно завершает работу без вызова **Метода BlockInput с параметром fBlock**, имеющим значение **FALSE**. В этом случае система выполняет правильную очистку и повторно включает входные данные.
- Пользователь нажимает клавиши **CTRL+ALT+DEL** или система вызывает модальное окно " **Жесткая системная ошибка** " (например, при сбое программы или сбое устройства).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

Основные понятия

[GetAsyncKeyState](#)

[GetKeyState](#)

[GetKeyboardState](#)

[Ввод с клавиатуры](#)

Справочные материалы

[SendInput](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция DefRawInputProc (winuser.h)

Статья 27.08.2023

В отличие от [DefWindowProcA](#) и [DefWindowProcW](#), эта функция не выполняет никакой обработки.

DefRawInputProc только проверяет, соответствует ли значение **cbSizeHeader** ожидаемому размеру [RAWINPUTHEADER](#).

Синтаксис

C++

```
LRESULT DefRawInputProc(
    [in] PRAWINPUT *paRawInput,
    [in] INT      nInput,
    [in] UINT     cbSizeHeader
);
```

Параметры

[in] paRawInput

Тип: **PRAWINPUT***

Не обрабатывается.

[in] nInput

Тип: **INT**

Не обрабатывается.

[in] cbSizeHeader

Тип: **UINT**

Размер структуры [RAWINPUTHEADER](#) (в байтах).

Возвращаемое значение

Тип: **LRESULT**

В случае успешного выполнения функция возвращает значение **0**. В противном случае возвращается значение **-1**.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-rawinput-l1-1-0 (появилось в Windows 10 версии 10.0.14393)

См. также раздел

[Основные понятия](#)

[RAWINPUT](#)

[RAWINPUTHEADER](#)

[Необработанные входные данные](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция DragDetect (winuser.h)

Статья 15.03.2023

Захватывает мышь и отслеживает ее движение, пока пользователь не отпустит левую кнопку мыши, не нажмет клавишу ESC или не переместит мышь за пределы прямоугольника перетаскивания, в котором находится указанная точка. Ширина и высота прямоугольника перетаскивания задаются значениями **SM_CXDRAG** и **SM_CYDRAG**, возвращаемыми функцией [GetSystemMetrics](#).

Синтаксис

C++

```
BOOL DragDetect(  
    [in] HWND hwnd,  
    [in] POINT pt  
>;
```

Параметры

[in] hwnd

Тип: **HWND**

Дескриптор для окна, получающего ввод с помощью мыши.

[in] pt

Тип: [POINT](#)

Начальное положение мыши в координатах экрана. Функция определяет координаты прямоугольника перетаскивания с помощью этой точки.

Возвращаемое значение

Тип: **BOOL**

Если пользователь переместил мышь за пределы прямоугольника перетаскивания, удерживая левую кнопку, возвращается ненулевое значение.

Если пользователь не переместил мышь за пределы прямоугольника перетаскивания, удерживая левую кнопку, возвращаемое значение равно нулю.

Комментарии

Системные метрики для прямоугольника перетаскивания настраиваются, что позволяет использовать большие или меньшие прямоугольники перетаскивания.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

Основные понятия

[GetSystemMetrics](#)

[Ввод с помощью мыши](#)

[ТОЧКИ](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция EnableWindow (winuser.h)

Статья 27.08.2023

Включает или отключает ввод с помощью мыши и клавиатуры для указанного окна или элемента управления. Если входные данные отключены, окно не получает такие входные данные, как щелчки мышью и нажатия клавиш. Если входные данные включены, окно получает все входные данные.

Синтаксис

C++

```
BOOL EnableWindow(  
    [in] HWND hWnd,  
    [in] BOOL bEnable  
>;
```

Параметры

[in] hWnd

Тип: **HWND**

Дескриптор окна для включения или отключения.

[in] bEnable

Тип: **BOOL**

Указывает, следует ли включать или отключать окно. Если этот параметр имеет значение **TRUE**, окно включено. Если параметр имеет значение **FALSE**, окно отключается.

Возвращаемое значение

Тип: **BOOL**

Если окно было ранее отключено, возвращаемое значение не равно нулю.

Если окно ранее не было отключено, возвращаемое значение равно нулю.

Комментарии

Если окно отключено, система отправляет [WM_CANCELMODE](#) сообщение. Если состояние включенного окна меняется, система отправляет [WM_ENABLE](#) сообщение после [WM_CANCELMODE](#) сообщения. (Эти сообщения отправляются до возврата [EnableWindow](#).) Если окно уже отключено, его дочерние окна неявно отключены, хотя они не отправляют [WM_ENABLE](#) сообщение.

Окно должно быть включено, прежде чем его можно будет активировать. Например, если приложение отображает немодерное диалоговое окно и отключило его main окно, приложение должно включить окно main перед уничтожением диалогового окна. В противном случае фокус клавиатуры будет активирован в другом окне. Если дочернее окно отключено, оно игнорируется, когда система пытается определить, какое окно должно получать сообщения мыши.

По умолчанию окно включено при его создании. Чтобы создать окно, которое изначально отключено, приложение может указать [стиль WS_DISABLED](#) в функции [CreateWindow](#) или [CreateWindowEx](#). После создания окна приложение может использовать [EnableWindow](#) для включения или отключения окна.

Приложение может использовать эту функцию для включения или отключения элемента управления в диалоговом окне. Отключенный элемент управления не может получить фокус клавиатуры, а пользователь не может получить к нему доступ.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

Набор API

ext-ms-win-ntuser-window-l1-1-4 (представлено в Windows 10 версии 10.0.14393)

См. также раздел

Основные понятия

[CreateWindow](#)

[CreateWindowEx](#)

[IsWindowEnabled](#)

[Ввод с клавиатуры](#)

Справочные материалы

[WM_ENABLE](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

макрос GET_APPCOMMAND_LPARAM (winuser.h)

Статья 27.08.2023

Извлекает команду приложения из указанного значения LPARAM .

Синтаксис

C++

```
void GET_APPCOMMAND_LPARAM(  
    lParam  
);
```

Параметры

lParam

Преобразуемое значение.

Возвращаемое значение

None

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

макрос GET_DEVICE_LPARAM (winuser.h)

Статья 01.07.2023

Извлекает тип устройства ввода из указанного значения LPARAM .

Синтаксис

C++

```
void GET_DEVICE_LPARAM(  
    lParam  
);
```

Параметры

lParam

Преобразуемое значение.

Возвращаемое значение

Возвращаемое значение — это бит слова высокого порядка, представляющего тип устройства ввода. Может быть одним из указанных далее.

Возвращаемый код/ значение	Описание
FAPPCOMMAND_KEY 0	Пользователь нажал клавишу.
0x8000	Пользователь нажал кнопку мыши.
FAPPCOMMAND_MOUSE	
0x1000	Неопознанный источник оборудования создал событие. Это
FAPPCOMMAND_OEM	может быть мышь или событие клавиатуры.

Возвращаемое значение

None

Remarks

Этот макрос идентичен [макросу GET_MOUSEORKEY_LPARAM](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

[макрос GET_MOUSEORKEY_LPARAM](#), ввод с помощью мыши

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

макрос GET_FLAGS_LPARAM (winuser.h)

Статья 27.08.2023

Извлекает состояние определенных виртуальных ключей из указанного значения LPARAM .

Синтаксис

C++

```
void GET_FLAGS_LPARAM(  
    lParam  
) ;
```

Параметры

lParam

Преобразуемое значение.

Возвращаемое значение

None

Remarks

Этот макрос идентичен [макросу GET_KEYSTATE_LPARAM](#) .

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows

См. также раздел

[Основные понятия](#)

[GET_KEYSTATE_LPARAM](#)

[Ввод с помощью мыши](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

макрос GET_KEYSTATE_LPARAM (winuser.h)

Статья 27.08.2023

Извлекает состояние определенных виртуальных ключей из указанного значения LPARAM .

Синтаксис

C++

```
void GET_KEYSTATE_LPARAM(  
    LPARAM  
) ;
```

Параметры

LPARAM

Преобразуемое значение.

Возвращаемое значение

None

Remarks

Этот макрос идентичен [макросу GET_FLAGS_LPARAM](#) .

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]

Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

[Основные понятия](#)

[GET_FLAGS_LPARAM](#)

[Ввод с помощью мыши](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

макрос GET_KEYSTATE_WPARAM (winuser.h)

Статья 27.08.2023

Извлекает состояние определенных виртуальных ключей из указанного значения WPARAM .

Синтаксис

C++

```
void GET_KEYSTATE_WPARAM(  
    wParam  
) ;
```

Параметры

wParam

Преобразуемое значение.

Возвращаемое значение

None

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

[Общие сведения о вводе с помощью мыши](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

макрос GET_NCHITTEST_WPARAM (winuser.h)

Статья 27.08.2023

Извлекает значение проверки попадания из указанного значения WPARAM .

Синтаксис

C++

```
void GET_NCHITTEST_WPARAM(  
    wParam  
);
```

Параметры

wParam

Преобразуемое значение.

Возвращаемое значение

None

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Макрос GET_RAWINPUT_CODE_WPARAM (winuser.h)

Статья 27.08.2023

Извлекает входной код из *wParam* в [WM_INPUT сообщении](#).

Синтаксис

C++

```
void GET_RAWINPUT_CODE_WPARAM(  
    wParam  
);
```

Параметры

wParam

wParam из [сообщения WM_INPUT](#).

Возвращаемое значение

Значение входного кода. Может применяться один из перечисленных ниже типов.

Значение	Значение
RIM_INPUT 0	Входные данные выполнялись, когда приложение находилось на переднем плане.
RIM_INPUTSINK 1	Входные данные выполнялись, когда приложение не находилось на переднем плане.

Требования

Минимальная версия клиента Windows XP [только классические приложения]

Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[RAWINPUT](#)

[Необработанные входные данные](#)

Справочные материалы

[WM_INPUT](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

макрос GET_WHEEL_DELTA_WPARAM (winuser.h)

Статья 27.08.2023

Извлекает значение wheel-delta из указанного значения WPARAM .

Синтаксис

C++

```
void GET_WHEEL_DELTA_WPARAM(  
    wParam  
);
```

Параметры

wParam

Преобразуемое значение.

Возвращаемое значение

None

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

макрос GET_XBUTTON_WPARAM (winuser.h)

Статья 27.08.2023

Извлекает состояние определенных кнопок из указанного значения **WPARAM**.

Синтаксис

C++

```
void GET_XBUTTON_WPARAM(  
    wParam  
);
```

Параметры

wParam

Преобразуемое значение.

Возвращаемое значение

None

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetActiveWindow (winuser.h)

Статья 27.08.2023

Извлекает дескриптор окна в активное окно, присоединенное к очереди сообщений вызывающего потока.

Синтаксис

C++

```
HWND GetActiveWindow();
```

Возвращаемое значение

Тип: **HWND**

Возвращаемое значение — это дескриптор активного окна, присоединенного к очереди сообщений вызывающего потока. В противном случае возвращаемое значение равно **NULL**.

Комментарии

Чтобы получить дескриптор в окне переднего плана, можно использовать [getForegroundWindow](#).

Чтобы получить дескриптор окна к активному окну в очереди сообщений для другого потока, используйте [GetGUIThreadInfo](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows

Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-window-l1-1-4 (появилась в Windows 10 версии 10.0.14393)

См. также раздел

Основные понятия

[GetForegroundWindow](#)

[GetGUIThreadInfo](#)

[Ввод с клавиатуры](#)

Справочные материалы

[SetActiveWindow](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetAsyncKeyState (winuser.h)

Статья 27.08.2023

Определяет, находится ли клавиша вверх или вниз во время вызова функции и была ли клавиша нажата после предыдущего вызова `GetAsyncKeyState`.

Синтаксис

C++

```
SHORT GetAsyncKeyState(  
    [in] int vKey  
);
```

Параметры

[in] vKey

Тип: `int`

Код виртуального ключа. Дополнительные сведения см. в разделе [Коды виртуальных ключей](#).

Для указания определенных ключей можно использовать различающиеся слева и справа константы. Дополнительные сведения см. в разделе Примечания.

Возвращаемое значение

Тип: `SHORT`

Если функция выполняется успешно, возвращаемое значение указывает, была ли клавиша нажата с момента последнего вызова `GetAsyncKeyState` и находится ли клавиша в данный момент вверх или вниз. Если задан самый значительный бит, клавиша не работает, а если задан наименьший значимый бит, клавиша была нажата после предыдущего вызова `GetAsyncKeyState`. Однако не следует полагаться на это последнее поведение; Дополнительные сведения см. в разделе Примечания.

Возвращаемое значение равно нулю в следующих случаях:

- Текущий рабочий стол не является активным

- Поток переднего плана принадлежит другому процессу, и рабочий стол не разрешает перехватчик или запись журнала.

Комментарии

Функция `GetAsyncKeyState` работает с кнопками мыши. Однако он проверяет состояние физических кнопок мыши, а не логических кнопок мыши, с которыми сопоставлены физические кнопки. Например, вызов `GetAsyncKeyState(VK_LBUTTON)` всегда возвращает состояние левой физической кнопки мыши, независимо от того, сопоставлена ли она с левой или правой логической кнопкой мыши. Вы можете определить текущее системное сопоставление физических кнопок мыши с логическими кнопками мыши, вызвав `.GetSystemMetrics(SM_SWAPBUTTON)`

возвращает значение `TRUE`, если кнопки мыши были перемещены.

Хотя наименьший значимый бит возвращаемого значения указывает, была ли клавиша нажата с момента последнего запроса, из-за упреждающего многозадачности Windows другое приложение может вызвать `GetAsyncKeyState` и получить бит "недавно нажат" вместо вашего приложения. Поведение наименьшего значимого бита возвращаемого значения сохраняется строго для совместимости с 16-разрядными приложениями Windows (которые не являются упреждаемыми) и не следует полагаться на.

Константы кода виртуального ключа можно использовать `VK_SHIFT`, `VK_CONTROL` и `VK_MENU` в качестве значений для параметра `vKey`. Это позволяет определить состояние клавиш SHIFT, CTRL или ALT без различия левого и правого.

Следующие константы кода виртуального ключа можно использовать в качестве значений для `vKey`, чтобы различать левый и правый экземпляры этих ключей.

Код	Значение
<code>VK_LSHIFT</code>	Клавиша shift влево.
<code>VK_RSHIFT</code>	Клавиша shift вправо.
<code>VK_LCONTROL</code>	Левая клавиша управления.
<code>VK_RCONTROL</code>	Клавиша правого элемента управления.
<code>VK_LMENU</code>	Клавиша меню слева.
<code>VK_RMENU</code>	Клавиша правого меню.

Эти различающиеся слева и справа константы доступны только при вызове функций [GetKeyboardState](#), [SetKeyboardState](#), [GetAsyncKeyState](#), [GetKeyState](#) и [MapVirtualKey](#).

Пример

C++

```
while (GetMessage(&msg, nullptr, 0, 0))
{
    if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    switch (msg.message)
    {
    case WM_KEYDOWN:
        if ((GetAsyncKeyState(VK_ESCAPE) & 0x01) && bRunning)
        {
            Stop();
        }
        break;
    }
}
```

Пример из [классических примеров Windows](#) на сайте GitHub.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
 - [GetKeyState](#)
 - [GetKeyboardState](#)
 - [GetSystemMetrics](#)
 - [MapVirtualKey](#)
 - [SetKeyboardState](#)
 - [Ввод с клавиатуры](#)
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetCapture (winuser.h)

Статья 27.08.2023

Извлекает дескриптор окна (если таковой имеется), захватив мышь. Только одно окно за раз может захватывать мышь; Это окно получает ввод мыши независимо от того, находится ли курсор в его границах.

Синтаксис

C++

```
HWND GetCapture();
```

Возвращаемое значение

Тип: HWND

Возвращаемое значение представляет собой дескриптор окна записи, связанного с текущим потоком. Если окно в потоке не захватило мышь, возвращаемое значение равно **NULL**.

Комментарии

Возвращаемое значение **NULL** означает, что текущий поток не захватил мышь. Однако мышь может быть захвачена другим потоком или процессом.

Чтобы получить дескриптор окна записи в другом потоке, используйте функцию [GetGUIThreadInfo](#) .

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows

Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-mouse-l1-1-0 (представлено в Windows 8)

См. также раздел

Основные понятия

[GetGUIThreadInfo](#)

[Ввод с помощью мыши](#)

Справочные материалы

[ReleaseCapture](#)

[SetCapture](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetDoubleClickTime (winuser.h)

Статья 27.08.2023

Извлекает текущее время двойного щелчка мыши. Двойной щелчок — это последовательность из двух щелчков кнопки мыши, второй — в течение указанного времени после первого. Время двойного щелчка — это максимальное количество миллисекунд, которое может произойти между первым и вторым щелчком двойного щелчка. Максимальное время двойного щелчка — 5000 миллисекунда.

Синтаксис

C++

```
UINT GetDoubleClickTime();
```

Возвращаемое значение

Тип: **UINT**

Возвращаемое значение указывает текущее время двойного щелчка в миллисекундах. Максимальное возвращаемое значение — 5000 миллисекунда.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с помощью мыши](#)

[Справочные материалы](#)

[SetDoubleClickTime](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetFocus (winuser.h)

Статья 27.08.2023

Извлекает дескриптор в окно с фокусом клавиатуры, если окно подключено к очереди сообщений вызывающего потока.

Синтаксис

C++

```
HWND GetFocus();
```

Возвращаемое значение

Тип: HWND

Возвращаемое значение — это дескриптор окна с фокусом клавиатуры. Если очередь сообщений вызывающего потока не имеет связанного окна с фокусом клавиатуры, возвращаемое значение равно **NULL**.

Комментарии

GetFocus возвращает окно с фокусом клавиатуры для очереди сообщений текущего потока. Если **GetFocus** возвращает значение **NULL**, очередь другого потока может быть присоединена к окну с фокусом клавиатуры.

Используйте функцию [GetForegroundWindow](#), чтобы получить дескриптор в окно, с которым в данный момент работает пользователь. Вы можете связать очередь сообщений потока с окнами, принадлежащими другому потоку, с помощью функции [AttachThreadInput](#).

Чтобы получить окно с фокусом клавиатуры на очереди переднего плана или очереди другого потока, используйте [функцию GetGUIThreadInfo](#).

Примеры

Пример см. в разделе "Создание панели инструментов поля со списком" [статьи Использование полей со списком](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-window-l1-1-4 (появилась в Windows 10 версии 10.0.14393)

См. также раздел

[AttachThreadInput](#)

Основные понятия

[GetForegroundWindow](#)

[GetGUIThreadInfo](#)

[Ввод с клавиатуры](#)

[Другие ресурсы](#)

Справочные материалы

[SetFocus](#)

[WM_KILLFOCUS](#)

[WM_SETFOCUS](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetKBCodePage (winuser.h)

Статья 27.08.2023

Извлекает текущую кодовую страницу.

Примечание Эта функция предоставляется только для совместимости с 16-разрядными версиями Windows. Приложения должны использовать функцию [GetOEMCP](#) для получения идентификатора кодовой страницы OEM для системы.

Синтаксис

C++

```
UINT GetKBCodePage();
```

Возвращаемое значение

Тип: **UINT**

Возвращаемое значение является идентификатором кодовой страницы OEM или идентификатором по умолчанию, если значение реестра недоступно для чтения. Список идентификаторов кодовой страницы изготовителя оборудования см. в разделе [Идентификаторы кодовых страниц](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

Библиотека	User32.lib
DLL	User32.dll

См. также раздел

Основные понятия

[GetACP](#)

[GetOEMCP](#)

[Ввод с клавиатуры](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetKeyboardLayout (winuser.h)

Статья 27.08.2023

Извлекает идентификатор активного языкового стандарта ввода (прежнее название — раскладка клавиатуры).

Синтаксис

C++

```
HKL GetKeyboardLayout(  
    [in] DWORD idThread  
);
```

Параметры

[in] idThread

Тип: **DWORD**

Идентификатор потока для запроса или 0 для текущего потока.

Возвращаемое значение

Тип: **HKL**

Возвращаемое значение — это идентификатор входного языкового стандарта для потока. Низкое слово содержит [идентификатор языка](#) ввода, а высокое — дескриптор устройства для физической раскладки клавиатуры.

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Так как раскладку клавиатуры можно динамически изменять, приложения, которые кэшируют сведения о текущей раскладке клавиатуры, должны обрабатывать

сообщение [WM_INPUTLANGCHANGE](#) , чтобы получать сведения об изменениях в языке ввода.

Чтобы получить KLID (идентификатор раскладки клавиатуры) текущего активного HKL, вызовите [GetKeyboardLayoutName](#).

Начиная с Windows 8: Предпочтительным методом для получения языка, связанного с текущей раскладкой клавиатуры или методом ввода, является вызов [Windows.Globalization.Language.CurrentInputMethodLanguageTag](#). Если приложение передает языковые теги из [CurrentInputMethodLanguageTag](#) в любые функции [поддержки национальных языков](#) , оно должно сначала преобразовать теги, вызвав [ResolveLocaleName](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[ActivateKeyboardLayout](#)

Основные понятия

[CreateThread](#)

[Ввод с клавиатуры](#)

[LoadKeyboardLayout](#)

Другие ресурсы

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetKeyboardLayoutList (winuser.h)

Статья 27.08.2023

Извлекает идентификаторы входного языкового стандарта (ранее называемые дескрипторами раскладки клавиатуры), соответствующие текущему набору входных языковых стандартов в системе. Функция копирует идентификаторы в указанный буфер.

Синтаксис

C++

```
int GetKeyboardLayoutList(
    [in] int nBuff,
    [out] HKL *lpList
);
```

Параметры

[in] nBuff

Тип: **int**

Максимальное количество дескрипторов, которые может содержать буфер.

[out] lpList

Тип: **HKL***

Указатель на буфер, получающий массив входных идентификаторов языкового стандарта.

Возвращаемое значение

Тип: **int**

Если функция выполняется успешно, возвращаемое значение — это количество идентификаторов входных языковых стандартов, скопированных в буфер, или, если *nBuff* равно нулю, возвращаемое значение — это размер буфера в элементах

массива, необходимый для получения всех текущих идентификаторов входных языковых стандартов.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Начиная с Windows 8: Предпочтительным методом для получения языка, связанного с текущей раскладкой клавиатуры или методом ввода, является вызов [Windows.Globalization.Language.CurrentInputMethodLanguageTag](#). Если приложение передает языковые теги из [CurrentInputMethodLanguageTag](#) в любые функции [поддержки национальных языков](#), оно должно сначала преобразовать теги, вызвав [ResolveLocaleName](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[GetKeyboardLayout](#)

[Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetKeyboardLayoutNameA (winuser.h)

Статья 21.07.2023

Извлекает имя идентификатора активного языкового стандарта ввода (прежнее название — раскладка клавиатуры) для вызывающего потока.

Синтаксис

C++

```
BOOL GetKeyboardLayoutNameA(  
    [out] LPSTR pwszKLID  
)
```

Параметры

[out] pwszKLID

Тип: LPTSTR

Буфер (длиной не менее KL_NAMELENGTH символов), который получает имя идентификатора входного языкового стандарта, включая завершающий символ NULL. Это будет копия строки, предоставленной функции [LoadKeyboardLayout](#), если не произошла подстановка макета.

Список входных макетов, которые поставляются вместе с Windows, см. в разделе [Идентификаторы клавиатуры и Редакторы методов ввода для Windows](#).

Возвращаемое значение

Тип: BOOL

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Начиная с Windows 8: Предпочтительным методом для получения языка, связанного с текущей раскладкой клавиатуры или методом ввода, является вызов `Windows.Globalization.Language.CurrentInputMethodLanguageTag`. Если приложение передает языковые теги из `CurrentInputMethodLanguageTag` в любые функции [поддержки национальных языков](#), оно должно сначала преобразовать теги, вызвав `ResolveLocaleName`.

ⓘ Примечание

Заголовок `winuser.h` определяет `GetKeyboardLayoutName` в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	<code>winuser.h</code> (включая <code>Windows.h</code>)
Библиотека	<code>User32.lib</code>
DLL	<code>User32.dll</code>

См. также раздел

[ActivateKeyboardLayout](#)

[Основные понятия](#)

[Ввод с клавиатуры](#)

[LoadKeyboardLayout](#)

[Справочные материалы](#)

[UnloadKeyboardLayout](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetKeyboardLayoutNameW (winuser.h)

Статья 21.07.2023

Извлекает имя идентификатора активного языкового стандарта ввода (прежнее название — раскладка клавиатуры) для вызывающего потока.

Синтаксис

C++

```
BOOL GetKeyboardLayoutNameW(  
    [out] LPWSTR pwszKLID  
) ;
```

Параметры

[out] pwszKLID

Тип: LPTSTR

Буфер (длиной не менее KL_NAMELENGTH символов), который получает имя идентификатора входного языкового стандарта, включая завершающий символ NULL. Это будет копия строки, предоставленной функции [LoadKeyboardLayout](#), если не произошла подстановка макета.

Список входных макетов, которые поставляются вместе с Windows, см. в разделе [Идентификаторы клавиатуры и Редакторы методов ввода для Windows](#).

Возвращаемое значение

Тип: BOOL

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Начиная с Windows 8: Предпочтительным методом для получения языка, связанного с текущей раскладкой клавиатуры или методом ввода, является вызов `Windows.Globalization.Language.CurrentInputMethodLanguageTag`. Если приложение передает языковые теги из `CurrentInputMethodLanguageTag` в любые функции [поддержки национальных языков](#), оно должно сначала преобразовать теги, вызвав `ResolveLocaleName`.

ⓘ Примечание

Заголовок `winuser.h` определяет `GetKeyboardLayoutName` в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	<code>winuser.h</code> (включая <code>Windows.h</code>)
Библиотека	<code>User32.lib</code>
DLL	<code>User32.dll</code>

См. также раздел

[ActivateKeyboardLayout](#)

[Основные понятия](#)

[Ввод с клавиатуры](#)

[LoadKeyboardLayout](#)

[Справочные материалы](#)

[UnloadKeyboardLayout](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetKeyboardState (winuser.h)

Статья 27.08.2023

Копирует состояние 256 виртуальных ключей в указанный буфер.

Синтаксис

C++

```
BOOL GetKeyboardState(  
    [out] PBYTE lpKeyState  
) ;
```

Параметры

[out] lpKeyState

Тип: **PBYTE**

256-байтовый массив, который получает данные о состоянии для каждого виртуального ключа.

Возвращаемое значение

Тип: **BOOL**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Приложение может вызвать эту функцию, чтобы получить текущее состояние всех виртуальных ключей. Состояние меняется, когда поток удаляет сообщения клавиатуры из очереди сообщений. Состояние не меняется, так как сообщения с клавиатуры отправляются в очередь сообщений потока и не изменяются по мере отправки сообщений с клавиатуры в очередь сообщений других потоков или извлечения из них. (Иключение: потоки, подключенные через [AttachThreadInput](#), имеют одно и то же состояние клавиатуры.)

При возврате функции каждый элемент массива, на который указывает параметр *lpKeyState*, содержит данные о состоянии виртуального ключа. Если бит высокого порядка равен 1, ключ не работает; в противном случае он работает. Если клавиша является переключателем, например CAPS LOCK, то бит низкого порядка равен 1, если ключ переключается, и равен 0, если ключ не переключен. Бит низкого порядка не имеет смысла для ключей без переключения. Говорят, что переключатель переключается, когда он включен. Индикатор клавиши переключателя (если таковой имеется) будет включен, когда клавиша включена, и выключается, когда клавиша отключена.

Чтобы получить сведения о состоянии отдельного ключа, используйте функцию [GetKeyState](#). Чтобы получить текущее состояние отдельной клавиши независимо от того, было ли получено соответствующее сообщение клавиатуры из очереди сообщений, используйте функцию [GetAsyncKeyState](#).

Приложение может использовать константы кода виртуального ключа **VK_SHIFT**, **VK_CONTROL** и **VK_MENU** в качестве индексов в массиве, на который указывает *lpKeyState*. Это позволяет определить состояние клавиш SHIFT, CTRL или ALT, не различая левое и правое. Приложение также может использовать следующие константы кода виртуального ключа в качестве индексов, чтобы различать левый и правый экземпляры этих ключей:

VK_LSHIFT
VK_RSHIFT
VK_LCONTROL
VK_RCONTROL
VK_LMENU
VK_RMENU

Эти различающиеся слева и справа константы доступны приложению только с помощью функций [GetKeyboardState](#), [SetKeyboardState](#), [GetAsyncKeyState](#), [GetKeyState](#) и [MapVirtualKey](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-rawinput-l1-1-0 (появилось в Windows 10 версии 10.0.14393)

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [GetSystemMetrics](#)
- [MapVirtualKey](#)
- [SetKeyboardState](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetKeyboardType (winuser.h)

Статья 27.08.2023

Извлекает сведения о текущей клавиатуре.

Синтаксис

C++

```
int GetKeyboardType(  
    [in] int nTypeFlag  
);
```

Параметры

[in] nTypeFlag

Тип: int

Тип извлекаемых сведений о клавиатуре. Этот параметр может принимать одно из указанных ниже значений.

Значение	Значение
0	Тип клавиатуры
1	Подтип клавиатуры
2	Количество функциональных клавиш на клавиатуре

Возвращаемое значение

Тип: int

Если функция выполняется успешно, возвращаемое значение указывает запрошенные сведения.

Если функция завершается сбоем и *nTypeFlag* не равно 1, возвращаемое значение равно 0; 0 является допустимым возвращаемым значением, если *nTypeFlag* равно 1 (подтип клавиатуры). Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Допустимые типы клавиатуры:

Значение	Описание
0x4	Улучшенные клавиатуры с 101 или 102 клавишами (и совместимые)
0x7	Японская клавиатура
0x8	Корейская клавиатура
0x51	Неизвестный тип или клавиатура HID

Подтипы клавиатуры — это значения, зависящие от изготовителя оборудования (OEM).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Функции ввода с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetKeyNameTextA (winuser.h)

Статья 15.11.2023

Извлекает строку, представляющую имя ключа.

Синтаксис

C++

```
int GetKeyNameTextA(
    [in] LONG lParam,
    [out] LPSTR lpString,
    [in] int cchSize
);
```

Параметры

[in] lParam

Тип: **LONG**

Второй параметр обрабатываемого сообщения клавиатуры (например, [WM_KEYDOWN](#)). Функция интерпретирует следующие позиции битов в *lParam*.

Bits	Значение
16—23	Код сканирования. Значение зависит от изготовителя оборудования.
24	Указывает, является ли клавиша расширенной, например клавишами ALT и CTRL справа, которые отображаются на клавиатуре с расширенными 101- или 102-клавишными клавишами. Значение равно 1, если это расширенный ключ; в противном случае — 0.
25	"Не волнует" бит. Приложение, вызывающее эту функцию, задает этот бит, чтобы указать, что функция не должна различать левую и правую клавиши CTRL и SHIFT, например.

Дополнительные сведения см. в разделе [Флаги сообщения нажатия](#) клавиш.

[out] lpString

Тип: **LPTSTR**

Буфер, который получит имя ключа.

[in] cchSize

Тип: int

Максимальная длина (в символах) имени ключа, включая завершающий символ NULL. (Этот параметр должен быть равен размеру буфера, на который указывает параметр *lpString*.)

Возвращаемое значение

Тип: int

Если функция выполняется успешно, строка, завершающаяся значением NULL, копируется в указанный буфер, а возвращаемое значение — это длина строки в символах, не подсчитывающая завершающий нуль-символ.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Формат строки "ключ-имя" зависит от текущей раскладки клавиатуры.

В раскладке клавиатуры хранится список имен в виде строк символов для клавиш с именами длиннее одного символа. Имя клавиши преобразуется в соответствии с [текущей активной раскладкой клавиатуры](#), поэтому функция может возвращать разные результаты для разных [раскладок клавиатуры](#).

Имя клавиши символа — это сам символ. Имена неуклюжих ключей приведены полностью.

Этот метод может неправильно работать с некоторыми [раскладками клавиатуры](#), которые создают несколько символов (т. е. лигатуры) и /или дополнительные символы Юникода, которые печатаются одним нажатием клавиши. Кроме того, ключи, сопоставленные с "A". [Коды виртуальных ключей](#) Z преобразуются в верхний регистр "A". Z' символы независимо от текущей раскладки клавиатуры. В таких случаях используйте [методы ToUnicode](#) или [ToUnicodeEx](#).

Заголовок winuser.h определяет GetKeyNameText как псевдоним, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Использование псевдонима, не

зависящий от кодирования, с кодом, который не является нейтральным для кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или времени выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Ввод с клавиатуры](#)

[Раскладки клавиатуры](#)

[Примеры раскладки клавиатуры](#)

[ToUnicode](#)

[ToUnicodeEx](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

Функция GetKeyNameTextW (winuser.h)

Статья 10.10.2023

Извлекает строку, представляющую имя ключа.

Синтаксис

C++

```
int GetKeyNameTextW(
    [in] LONG lParam,
    [out] LPWSTR lpString,
    [in] int cchSize
);
```

Параметры

[in] lParam

Тип: **LONG**

Второй параметр обрабатываемого сообщения клавиатуры (например, [WM_KEYDOWN](#)). Функция интерпретирует следующие позиции битов в *lParam*.

Bits	Значение
16—23	Код сканирования. Значение зависит от изготовителя оборудования.
24	Указывает, является ли клавиша расширенной, например клавишами ALT и CTRL справа, которые отображаются на клавиатуре с расширенными 101- или 102-клавишными клавишами. Значение равно 1, если это расширенный ключ; в противном случае — 0.
25	"Не волнует" бит. Приложение, вызывающее эту функцию, задает этот бит, чтобы указать, что функция не должна различать левую и правую клавиши CTRL и SHIFT, например.

Дополнительные сведения см. в разделе [Флаги сообщения нажатия](#) клавиш.

[out] lpString

Тип: **LPTSTR**

Буфер, который получит имя ключа.

[in] cchSize

Тип: int

Максимальная длина (в символах) имени ключа, включая завершающий символ NULL. (Этот параметр должен быть равен размеру буфера, на который указывает параметр *lpString*.)

Возвращаемое значение

Тип: int

Если функция выполняется успешно, строка, завершающаяся значением NULL, копируется в указанный буфер, а возвращаемое значение — это длина строки в символах, не подсчитывающая завершающий нуль-символ.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Формат строки "ключ-имя" зависит от текущей раскладки клавиатуры.

В раскладке клавиатуры хранится список имен в виде строк символов для клавиш с именами длиннее одного символа. Имя клавиши преобразуется в соответствии с [текущей активной раскладкой клавиатуры](#), поэтому функция может возвращать разные результаты для разных [раскладок клавиатуры](#).

Имя клавиши символа — это сам символ. Имена неуклюжих ключей приведены полностью.

Этот метод может неправильно работать с некоторыми [раскладками клавиатуры](#), которые создают несколько символов (т. е. лигатуры) и /или дополнительные символы Юникода, которые печатаются одним нажатием клавиши. Кроме того, ключи, сопоставленные с "A". [Коды виртуальных ключей](#) Z преобразуются в верхний регистр "A". Z' символы независимо от текущей раскладки клавиатуры. В таких случаях используйте [методы ToUnicode](#) или [ToUnicodeEx](#).

Заголовок winuser.h определяет GetKeyNameText как псевдоним, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Использование псевдонима, не

зависящий от кодирования, с кодом, который не является нейтральным для кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или времени выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Ввод с клавиатуры](#)

[Раскладки клавиатуры](#)

[Примеры раскладки клавиатуры](#)

[ToUnicode](#)

[ToUnicodeEx](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

Функция GetKeyState (winuser.h)

Статья 27.08.2023

Извлекает состояние указанного виртуального ключа. Состояние указывает, находится ли клавиша вверх, вниз или переключается (вкл., выключается— чередуется каждый раз при нажатии клавиши).

Синтаксис

C++

```
SHORT GetKeyState(  
    [in] int nVirtKey  
) ;
```

Параметры

[in] nVirtKey

Тип: int

Виртуальный ключ. Если нужным виртуальным ключом является буква или цифра (от A до Z, от a до z или от 0 до 9), для *nVirtKey* должно быть задано значение ASCII этого символа. Для других ключей это должен быть код виртуального ключа.

Если используется раскладка клавиатуры, отличной от английского, для указания большинства символьных клавиш используются виртуальные клавиши со значениями в диапазоне ASCII A–Z и от 0 до 9. Например, для немецкой раскладки клавиатуры виртуальная клавиша со значением ASCII O (0x4F) относится к клавише "o", тогда как VK_OEM_1 относится к клавише "o with umlaut".

Возвращаемое значение

Тип: SHORT

Возвращаемое значение указывает состояние указанного виртуального ключа следующим образом:

- Если бит высокого порядка равен 1, ключ не работает; в противном случае он работает.

- Если бит нижнего порядка равен 1, ключ переключается. Если клавиша CAPS LOCK включена, она переключается. Ключ отключен и отключается, если бит низкого порядка равен 0. Индикатор клавиши переключателя (если таковой имеется) будет включен, когда клавиша включена, и выключается, когда клавиша отключена.

Комментарии

Состояние ключа, возвращаемого этой функцией, изменяется по мере того, как поток считывает ключевые сообщения из своей очереди сообщений. Состояние не отражает состояние уровня прерывания, связанное с оборудованием. Используйте функцию [GetAsyncKeyState](#) для получения этой информации.

Приложение вызывает [GetKeyState](#) в ответ на сообщение ввода с клавиатуры. Эта функция извлекает состояние ключа при создании входного сообщения.

Чтобы получить сведения о состоянии для всех виртуальных ключей, используйте функцию [GetKeyboardState](#).

Приложение может использовать константы [кода виртуального ключа](#) **VK_SHIFT**, **VK_CONTROL** и **VK_MENU** в качестве значений для параметра *nVirtKey*. Это позволяет определить состояние клавиш SHIFT, CTRL или ALT, не различая левое и правое. Приложение также может использовать следующие константы кода виртуального ключа в качестве значений для *nVirtKey*, чтобы различать левый и правый экземпляры этих ключей:

VK_LSHIFT **VK_RSHIFT** **VK_LCONTROL** **VK_RCONTROL** **VK_LMENU** **VK_RMENU** Эти различающиеся слева и справа константы доступны приложению только с помощью функций [GetKeyboardState](#), [SetKeyboardState](#), [GetAsyncKeyState](#), [GetKeyState](#) и [MapVirtualKey](#).

Примеры

Пример см. в разделе [Отображение ввода с клавиатуры](#).

Требования

Минимальная версия
клиента

Windows 2000 Professional [только классические
приложения]

Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [MapVirtualKey](#)
- [SetKeyboardState](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetLastInputInfo (winuser.h)

Статья 27.08.2023

Извлекает время последнего входного события.

Синтаксис

C++

```
BOOL GetLastInputInfo(  
    [out] PLASTINPUTINFO plii  
) ;
```

Параметры

[out] plii

Тип: **PLASTINPUTINFO**

Указатель на структуру [LASTINPUTINFO](#), которая получает время последнего события ввода.

Возвращаемое значение

Тип: **BOOL**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Комментарии

Эта функция полезна для обнаружения бездействия входных данных. Однако **GetLastInputInfo** не предоставляет системные сведения о входных данных пользователя во всех запущенных сеансах. Вместо этого **GetLastInputInfo** предоставляет сведения о входных данных пользователя для конкретного сеанса только для сеанса, который вызвал функцию.

Количество тактов при получении последнего входного события (см. [LastINPUTINFO](#)) не гарантируется приращением. В некоторых случаях значение

может быть меньше числа тактов предыдущего события. Например, это может быть вызвано разрывом времени между необработанным входным потоком и потоком рабочего стола или событием, вызванным [SendInput](#), который предоставляет собственный счетчик тактов.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с клавиатуры](#)

[LASTINPUTINFO](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetMouseMovePointsEx (winuser.h)

Статья 27.08.2023

Извлекает журнал до 64 предыдущих координат мыши или пера.

Синтаксис

C++

```
int GetMouseMovePointsEx(
    [in]  UINT          cbSize,
    [in]  LPMOUSEMOVEPOINT lppt,
    [out] LPMOUSEMOVEPOINT lpptBuf,
    [in]  int           nBufPoints,
    [in]  DWORD         resolution
);
```

Параметры

[in] cbSize

Тип: **UINT**

Размер структуры **MOUSEMOVEPOINT** в байтах.

[in] lppt

Тип: **LPMOUSEMOVEPOINT**

Указатель на структуру **MOUSEMOVEPOINT**, содержащую допустимые координаты мыши (в экранных координатах). Он также может содержать метку времени.

Функция **GetMouseMovePointsEx** выполняет поиск точки в журнале координат мыши. Если функция находит точку, она возвращает последние значения *nBufPoints* до указанной точки и включая ее.

Если приложение предоставляет метку времени, функция **GetMouseMovePointsEx** будет использовать ее для различия двух равных точек, записанных в разное время.

Приложение должно вызвать эту функцию, используя координаты мыши, полученные из [сообщения WM_MOUSEMOVE](#), и преобразовать их в экранные координаты.

[out] lpptBuf

Тип: **LPMOUSEMOVEPOINT**

Указатель на буфер, который будет получать точки. Размер должен быть не менее $cbSize * nBufPoints$.

[in] nBufPoints

Тип: **int**

Количество извлекаемых точек.

[in] resolution

Тип: **DWORD**

Требуемое разрешение. Этот параметр может принимать одно из указанных ниже значений.

Значение	Значение
GMMP_USE_DISPLAY_POINTS 1	Извлекает точки с помощью разрешения экрана.
GMMP_USE_HIGH_RESOLUTION_POINTS 2	Извлекает точки с высоким разрешением. Точки могут варьироваться от нуля до 65 535 (0xFFFF) в координатах X и Y. Это разрешение обеспечивается устройствами, указывающими на абсолютные координаты, такими как планшеты для рисования.

Возвращаемое значение

Тип: **int**

Если функция выполняется успешно, возвращаемое значение — это количество точек в буфере. В противном случае функция возвращает -1. Для получения дополнительных сведений об ошибке приложение может вызывать [GetLastError](#).

Комментарии

Система сохраняет последние 64 координаты мыши и их метки времени. Если приложение предоставляет координату мыши для **GetMouseMovePointsEx** и координата существует в журнале координат мыши системы, функция извлекает указанное количество координат из журнала систем. Можно также указать метку времени, которая будет использоваться для различения идентичных точек в журнале.

Функция **GetMouseMovePointsEx** возвращает точки, которые в конечном итоге были отправлены не только в вызывающий поток, но и в другие потоки.

GetMouseMovePointsEx может завершаться ошибкой или возвращать ошибочные значения в следующих случаях:

- Значение , если отрицательные координаты передаются в структуре [MOUSEMOVEPOINT](#) .
- Если **GetMouseMovePointsEx** получает координату с отрицательным значением.

Такие ситуации могут возникать при наличии нескольких мониторов. Чтобы исправить это, сначала вызовите [GetSystemMetrics](#) , чтобы получить следующие значения:

- **SM_XVIRTUALSCREEN**,
- **SM_YVIRTUALSCREEN**,
- **SM_CXVIRTUALSCREEN** и
- **SM_CYVIRTUALSCREEN**.

Затем для каждой точки, возвращаемой из **GetMouseMovePointsEx**, выполните следующее преобразование:

```
int nVirtualWidth = GetSystemMetrics(SM_CXVIRTUALSCREEN) ;
int nVirtualHeight = GetSystemMetrics(SM_CYVIRTUALSCREEN) ;
int nVirtualLeft = GetSystemMetrics(SM_XVIRTUALSCREEN) ;
int nVirtualTop = GetSystemMetrics(SM_YVIRTUALSCREEN) ;
int cpt = 0 ;
int mode = GMMP_USE_DISPLAY_POINTS ;

MOUSEMOVEPOINT mp_in ;
MOUSEMOVEPOINT mp_out[64] ;

ZeroMemory(&mp_in, sizeof(mp_in)) ;
mp_in.x = pt.x & 0x0000FFFF ;//Ensure that this number will pass through.
mp_in.y = pt.y & 0x0000FFFF ;
cpt = GetMouseMovePointsEx(&mp_in, &mp_out, 64, mode) ;
```

```

for (int i = 0; i < cpt; i++)
{
    switch(mode)
    {
        case GMMP_USE_DISPLAY_POINTS:
            if (mp_out[i].x > 32767)
                mp_out[i].x -= 65536 ;
            if (mp_out[i].y > 32767)
                mp_out[i].y -= 65536 ;
            break ;
        case GMMP_USE_HIGH_RESOLUTION_POINTS:
            mp_out[i].x = ((mp_out[i].x * (nVirtualWidth - 1)) - (nVirtualLeft * 65536)) / nVirtualWidth ;
            mp_out[i].y = ((mp_out[i].y * (nVirtualHeight - 1)) - (nVirtualTop * 65536)) / nVirtualHeight ;
            break ;
    }
}

```

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[MOUSEMOVEPOINT](#)

[Ввод с помощью мыши](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetRawInputBuffer (winuser.h)

Статья 27.08.2023

Выполняет буферизованное чтение необработанных входных данных сообщений, найденных в очереди сообщений вызывающего потока.

Синтаксис

C++

```
UINT GetRawInputBuffer(
    [out, optional] PRAWINPUT pData,
    [in, out]        PUINT     pcbSize,
    [in]             UINT      cbSizeHeader
);
```

Параметры

[out, optional] pData

Тип: **PRAWINPUT**

Указатель на буфер структур **RAWINPUT**, содержащих необработанные входные данные. Буфер должен быть выровнен по границе указателя, которая является **DWORD** для 32-разрядных архитектур и **QWORD** для 64-разрядных архитектур.

Если значение **PABHO NULL**, размер первых необработанных входных данных сообщения (минимальный необходимый буфер) в байтах возвращается в **pcbSize*.

[in, out] pcbSize

Тип: **PUINT**

Размер предоставленного буфера **RAWINPUT** в байтах.

[in] cbSizeHeader

Тип: **UINT**

Размер структуры **RAWINPUTHEADER** в байтах.

Возвращаемое значение

Тип: **UINT**

Если *pData* имеет значение **NULL** и функция выполнена успешно, возвращаемое значение равно нулю. Если значение *pData* не равно **NULL** и функция выполнена успешно, возвращаемое значение — это количество структур **RAWINPUT**, записанных в *pData*.

При возникновении ошибки возвращается значение (**UINT**)-1. Вызовите [GetLastError](#) для кода ошибки.

Комментарии

Когда приложение получает необработанные входные данные, его очередь сообщений получает [WM_INPUT](#) сообщение и флаг состояния очереди [QS_RAWINPUT](#) установлен.

С помощью [GetRawInputBuffer](#) необработанные входные данныечитываются в массиве структур **RAWINPUT** переменного размера, а соответствующие [сообщения WM_INPUT](#) удаляются из очереди сообщений вызывающего потока. Этот метод можно вызвать несколько раз с буфером, который не может поместить все данные сообщения, пока не будут прочитаны все необработанные входные сообщения.

Макрос [NEXTRAWINPUTBLOCK](#) позволяет приложению просматривать массив структур **RAWINPUT**.

Если все необработанные входные сообщения успешно считаются из очереди сообщений, [QS_RAWINPUT](#) флаг удаляется из состояния очереди сообщений вызывающего потока.

ⓘ Примечание

WOW64: чтобы получить правильный размер необработанного входного буфера, не используйте **pcbSize*, вместо него используйте **pcbSize * 8*. Чтобы обеспечить правильное поведение [GetRawInputBuffer](#) в WOW64, необходимо выровнять структуру **RAWINPUT** по 8 байтам. В следующем коде показано, как выровнять **RAWINPUT** для WOW64.

C#

```
[StructLayout(LayoutKind.Explicit)]
internal struct RAWINPUT
{
    [FieldOffset(0)]
```

```
public RAWINPUTHEADER header;  
  
[FieldOffset(16+8)]  
public RAWMOUSE mouse;  
  
[FieldOffset(16+8)]  
public RAWKEYBOARD keyboard;  
  
[FieldOffset(16+8)]  
public RAWHID hid;  
}
```

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[GetMessage](#)

[NEXTRAWINPUTBLOCK](#)

[RAWINPUT](#)

[RAWINPUTHEADER](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

[Общие сведения о необработанных входных данных](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetRawInputData (winuser.h)

Статья 27.08.2023

Извлекает необработанные входные данные с указанного устройства.

Синтаксис

C++

```
UINT GetRawInputData(
    [in]          HRAWINPUT hRawInput,
    [in]          UINT      uiCommand,
    [out, optional] LPVOID   pData,
    [in, out]     PUINT    pcbSize,
    [in]          UINT      cbSizeHeader
);
```

Параметры

[in] hRawInput

Тип: **HRAWINPUT**

Дескриптор структуры [RAWINPUT](#). Это происходит от *lParam* в [WM_INPUT](#).

[in] uiCommand

Тип: **UINT**

Флаг команды. Этот параметр может принимать одно из указанных ниже значений.

Значение	Значение
RID_HEADER 0x10000005	Получение сведений о заголовке из структуры RAWINPUT .
RID_INPUT 0x10000003	Получение необработанных данных из структуры RAWINPUT .

[out, optional] pData

Тип: **LPVOID**

Указатель на данные, поступающие из структуры [RAWINPUT](#). Это зависит от значения *uiCommand*. Если *pData* имеет значение **NULL**, требуемый размер буфера возвращается в **pcbSize*.

[in, out] *pcbSize*

Тип: **PUINT**

Размер данных в байтах в *pData*.

[in] *cbSizeHeader*

Тип: **UINT**

Размер структуры [RAWINPUTHEADER](#) в байтах.

Возвращаемое значение

Тип: **UINT**

Если *pData* имеет значение **NULL** и функция выполнена успешно, возвращаемое значение равно 0. Если значение *pData* не равно **NULL** и функция выполнена успешно, возвращаемое значение — это количество байтов, скопированных в *pData*.

При возникновении ошибки возвращается значение **(UINT)-1**.

Комментарии

[GetRawInputData](#) получает необработанные входные данные по одной структуре [RAWINPUT](#) за раз. В отличие от этого, [GetRawInputBuffer](#) получает массив структур [RAWINPUT](#).

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows

Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-rawinput-l1-1-0 (представлено в Windows 10 версии 10.0.14393)

См. также раздел

Основные понятия

[GetRawInputBuffer](#)

[RAWINPUT](#)

[RAWINPUTHEADER](#)

[Необработанные входные данные](#)

Справочные материалы

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetRawInputDeviceInfoA (winuser.h)

Статья 27.08.2023

Извлекает сведения о необработанном устройстве ввода.

Синтаксис

C++

```
UINT GetRawInputDeviceInfoA(
    [in, optional]     HANDLE hDevice,
    [in]                UINT   uiCommand,
    [in, out, optional] LPVOID pData,
    [in, out]           PUINT pcbSize
);
```

Параметры

[in, optional] hDevice

Тип: **HANDLE**

Дескриптор для необработанного устройства ввода. Это происходит от члена **hDevice** [RAWINPUTHEADER](#) или [getRawInputDeviceList](#).

[in] uiCommand

Тип: **UINT**

Указывает, какие данные будут возвращены в *pData*. Этот параметр может принимать одно из указанных ниже значений.

Значение	Значение
RIDI_PREPARSEDDATA 0x20000005	<i>pData</i> — это PHIDP_PREPARSED_DATA указатель на буфер для подготовленных данных коллекции верхнего уровня .
RIDI_DEVICENAME 0x20000007	<i>pData</i> указывает на строку, содержащую имя интерфейса устройства .

Если это устройство [открыто в режиме общего доступа](#), вы можете вызвать [CreateFile](#) с этим именем, чтобы открыть коллекцию HID и использовать возвращенный дескриптор для вызова [ReadFile](#) для чтения входных отчетов и [WriteFile](#) для отправки выходных отчетов.

Дополнительные сведения см. в разделе [Открытие коллекций HID](#) и [обработка отчетов HID](#).

Только для этого *uiCommand* значением в *pcbSize* является число символов (а не число байтов).

RIDI_DEVICEINFO

0x2000000b

pData указывает на [структуру RIDI_DEVICE_INFO](#).

`[in, out, optional] pData`

Тип: **LPVOID**

Указатель на буфер, содержащий сведения, заданные *uiCommand*.

Если *uiCommand* имеет значение **RIDI_DEVICEINFO**, перед вызовом [GetRawInputDeviceInfo](#) задайте для члена *cbSize* `RIDI_DEVICE_INFO sizeof(RIDI_DEVICE_INFO)` значение .

`[in, out] pcbSize`

Тип: **PUINT**

Размер данных в *pData* (в байтах).

Возвращаемое значение

Тип: **UINT**

В случае успешного выполнения эта функция возвращает не отрицательное число, указывающее количество байтов, скопированных в *pData*.

Если значение *pData* недостаточно велико для данных, функция возвращает значение -1. Если *pData* имеет значение **NULL**, функция возвращает нулевое значение. В обоих случаях *pcbSize* имеет минимальный размер, необходимый для буфера *pData*.

Вызовите [Метод GetLastError](#), чтобы определить любые другие ошибки.

Комментарии

ⓘ Примечание

Заголовок winuser.h определяет GetRawInputDeviceInfo в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-rawinput-l1-1-0 (появилось в Windows 10 версии 10.0.14393)

См. также раздел

[Основные понятия](#)

[RAWINPUTHEADER](#)

[RID_DEVICE_INFO](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

WM_INPUT

Коллекции верхнего уровня

Предварительно подготовленные данные

PHIDP_PREPARSED_DATA

Открытие коллекций HID

Обработка отчетов HID

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

Получить справку в Microsoft Q&A

Функция GetRawInputDeviceInfoW (winuser.h)

Статья 27.08.2023

Извлекает сведения о необработанном устройстве ввода.

Синтаксис

C++

```
UINT GetRawInputDeviceInfoW(
    [in, optional]     HANDLE hDevice,
    [in]                UINT   uiCommand,
    [in, out, optional] LPVOID pData,
    [in, out]           PUINT pcbSize
);
```

Параметры

[in, optional] hDevice

Тип: **HANDLE**

Дескриптор для необработанного устройства ввода. Это происходит от члена **hDevice** [RAWINPUTHEADER](#) или [getRawInputDeviceList](#).

[in] uiCommand

Тип: **UINT**

Указывает, какие данные будут возвращены в *pData*. Этот параметр может принимать одно из указанных ниже значений.

Значение	Значение
RIDI_PREPARSEDDATA 0x20000005	<i>pData</i> — это PHIDP_PREPARSED_DATA указатель на буфер для подготовленных данных коллекции верхнего уровня .
RIDI_DEVICENAME 0x20000007	<i>pData</i> указывает на строку, содержащую имя интерфейса устройства .

Если это устройство [открыто в режиме общего доступа](#), вы можете вызвать [CreateFile](#) с этим именем, чтобы открыть коллекцию HID и использовать возвращенный дескриптор для вызова [ReadFile](#) для чтения входных отчетов и [WriteFile](#) для отправки выходных отчетов.

Дополнительные сведения см. в разделе [Открытие коллекций HID](#) и [обработка отчетов HID](#).

Только для этого *uiCommand* значением в *pcbSize* является число символов (а не число байтов).

RIDI_DEVICEINFO

0x2000000b

pData указывает на [структуру RIDI_DEVICE_INFO](#).

`[in, out, optional] pData`

Тип: **LPVOID**

Указатель на буфер, содержащий сведения, заданные *uiCommand*.

Если *uiCommand* имеет значение **RIDI_DEVICEINFO**, перед вызовом [GetRawInputDeviceInfo](#) задайте для члена *cbSize* `RIDI_DEVICE_INFO sizeof(RIDI_DEVICE_INFO)` значение .

`[in, out] pcbSize`

Тип: **PUINT**

Размер данных в *pData* (в байтах).

Возвращаемое значение

Тип: **UINT**

В случае успешного выполнения эта функция возвращает не отрицательное число, указывающее количество байтов, скопированных в *pData*.

Если значение *pData* недостаточно велико для данных, функция возвращает значение -1. Если *pData* имеет значение **NULL**, функция возвращает нулевое значение. В обоих случаях *pcbSize* имеет минимальный размер, необходимый для буфера *pData*.

Вызовите [Метод GetLastError](#), чтобы определить любые другие ошибки.

Комментарии

ⓘ Примечание

Заголовок winuser.h определяет GetRawInputDeviceInfo в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-rawinput-l1-1-0 (появилось в Windows 10 версии 10.0.14393)

См. также раздел

[Основные понятия](#)

[RAWINPUTHEADER](#)

[RID_DEVICE_INFO](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

WM_INPUT

Коллекции верхнего уровня

Предварительно подготовленные данные

PHIDP_PREPARSED_DATA

Открытие коллекций HID

Обработка отчетов HID

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

Получить справку в Microsoft Q&A

Функция GetRawInputDeviceList (winuser.h)

Статья 27.08.2023

Перечисляет необработанные устройства ввода, подключенные к системе.

Синтаксис

C++

```
UINT GetRawInputDeviceList(
    [out, optional] PRAWINPUTDEVICELIST pRawInputDeviceList,
    [in, out]        PUINT             puiNumDevices,
    [in]              UINT              cbSize
);
```

Параметры

[out, optional] pRawInputDeviceList

Тип: **PRAWINPUTDEVICELIST**

Массив структур **RAWINPUTDEVICELIST** для устройств, подключенных к системе.

Если значение РАВНО **NULL**, количество устройств возвращается в ***puiNumDevices**.

[in, out] puiNumDevices

Тип: **PUINT**

Если **pRawInputDeviceList** имеет значение **NULL**, функция заполняет эту переменную количеством устройств, подключенных к системе; В противном случае эта переменная указывает количество структур **RAWINPUTDEVICELIST**, которые могут содержаться в буфере, к которому указывает **pRawInputDeviceList**. Если это значение меньше числа устройств, подключенных к системе, функция возвращает фактическое количество устройств в этой переменной и завершается сбоем с **ERROR_INSUFFICIENT_BUFFER**. Если это значение больше или равно количеству устройств, подключенных к системе, значение не изменяется, а число устройств указывается в качестве возвращаемого значения.

[in] cbSize

Тип: **UINT**

Размер структуры **RAWINPUTDEVICELIST** в байтах.

Возвращаемое значение

Тип: **UINT**

Если функция выполнена успешно, возвращается количество устройств, хранящихся в буфере, на который указывает *pRawInputDeviceList*.

При любой другой ошибке функция возвращает (**UINT**) -1, а [GetLastError](#) возвращает указание об ошибке.

Комментарии

Из этой функции возвращаются мышь, клавиатура и другие устройства HID.

Чтобы получить более подробные сведения о подключенных устройствах, вызовите [Метод GetRawInputDeviceInfo](#) с помощью *hDevice* из **RAWINPUTDEVICELIST**.

Примеры

В следующем примере кода показан типичный вызов [GetRawInputDeviceList](#):

C++

```
UINT nDevices;
PRAWINPUTDEVICELIST pRawInputDeviceList = NULL;
while (true) {
    if (GetRawInputDeviceList(NULL, &nDevices, sizeof(RAWINPUTDEVICELIST))
!= 0) { Error(); }
    if (nDevices == 0) { break; }
    if ((pRawInputDeviceList = malloc(sizeof(RAWINPUTDEVICELIST) *
nDevices)) == NULL) { Error(); }
    nDevices = GetRawInputDeviceList(pRawInputDeviceList, &nDevices,
sizeof(RAWINPUTDEVICELIST));
    if (nDevices == (UINT)-1) {
        if (GetLastError() != ERROR_INSUFFICIENT_BUFFER) { Error(); }
        // Devices were added.
        free(pRawInputDeviceList);
        continue;
    }
    break;
}
// do the job...
```

```
// after the job, free the RAWINPUTDEVICELIST  
free(pRawInputDeviceList);
```

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-rawinput-l1-1-0 (появилось в Windows 10 версии 10.0.14393)

См. также раздел

[Основные понятия](#)

[GetRawInputDeviceInfo](#)

[RAWINPUTDEVICELIST](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция GetRegisteredRawInputDevices (winuser.h)

Статья 27.08.2023

Извлекает сведения о необработанных устройствах ввода для текущего приложения.

Синтаксис

C++

```
UINT GetRegisteredRawInputDevices(
    [out, optional] PRAWINPUTDEVICE pRawInputDevices,
    [in, out]         PUINT          puiNumDevices,
    [in]              UINT           cbSize
);
```

Параметры

[out, optional] pRawInputDevices

Тип: **PRAWINPUTDEVICE**

Массив структур **RAWINPUTDEVICE** для приложения.

[in, out] puiNumDevices

Тип: **PUINT**

Количество структур **RAWINPUTDEVICE** в *pRawInputDevices.

[in] cbSize

Тип: **UINT**

Размер структуры **RAWINPUTDEVICE** в байтах.

Возвращаемое значение

Тип: **UINT**

В случае успешного выполнения функция возвращает не отрицательное число, которое является числом структур **RAWINPUTDEVICE**, записанных в буфер.

Если буфер *pRawInputDevices* слишком мал или имеет значение **NULL**, функция задает последнюю ошибку как **ERROR_INSUFFICIENT_BUFFER**, возвращает значение -1 и задает для *riNumDevices* необходимое количество устройств. Если функция завершается сбоем по какой-либо другой причине, она возвращает значение -1. Для получения дополнительных сведений вызовите [Метод GetLastError](#).

Комментарии

Чтобы получить необработанные входные данные с устройства, приложение должно зарегистрировать его с помощью [RegisterRawInputDevices](#).

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[RAWINPUTDEVICE](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

[RegisterRawInputDevices](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура HARDWAREINPUT (winuser.h)

Статья 28.08.2023

Содержит сведения о имитированном сообщении, созданном с помощью устройства ввода, отличного от клавиатуры или мыши.

Синтаксис

C++

```
typedef struct tagHARDWAREINPUT {
    DWORD uMsg;
    WORD wParamL;
    WORD wParamH;
} HARDWAREINPUT, *PHARDWAREINPUT, *LPHARDWAREINPUT;
```

Члены

uMsg

Тип: **DWORD**

Сообщение, созданное оборудованием ввода.

wParamL

Тип: **WORD**

Слово нижнего порядка параметра *lParam* для **uMsg**.

wParamH

Тип: **WORD**

Слово высокого порядка параметра *lParam* для **uMsg**.

Требования

Минимальная версия клиента

Windows XP [только классические приложения]

Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[INPUT](#)

[Ввод с клавиатуры](#)

[Справочные материалы](#)

[SendInput](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура INPUT (winuser.h)

Статья 28.08.2023

Используется [SendInput](#) для хранения информации для синтеза событий ввода, таких как нажатия клавиш, перемещение мыши и щелчки мышью.

Синтаксис

C++

```
typedef struct tagINPUT {
    DWORD type;
    union {
        MOUSEINPUT     mi;
        KEYBDINPUT    ki;
        HARDWAREINPUT hi;
    } DUMMYUNIONNAME;
} INPUT, *PINPUT, *LPIINPUT;
```

Члены

type

Тип: **DWORD**

Тип входного события. Этот элемент может быть одним из следующих значений.

Значение	Значение
INPUT_MOUSE 0	Событие является событием мыши. Используйте структуру mi для объединения.
INPUT_KEYBOARD 1	Событие является событием клавиатуры. Используйте структуру ki объединения.
INPUT_HARDWARE 2	Событие является событием оборудования. Используйте структуру hi объединения.

DUMMYUNIONNAME

DUMMYUNIONNAME.mi

Тип: [MOUSEINPUT](#)

Сведения о имитированном событии мыши.

DUMMYUNIONNAME.ki

Тип: [KEYBDINPUT](#)

Сведения о событии имитации клавиатуры.

DUMMYUNIONNAME.hi

Тип: [HARDWAREINPUT](#)

Сведения о имитированном событии оборудования.

Комментарии

INPUT_KEYBOARD поддерживает неключевые методы ввода, такие как распознавание рукописного ввода или распознавание голоса, как если бы это был ввод текста с помощью флага **KEYEVENTF_UNICODE**. Дополнительные сведения см. в разделе примечаний [статьи KEYBDINPUT](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[GetMessageExtraInfo](#)

[HARDWAREINPUT](#)

[KEYBDINPUT](#)

[Ввод с клавиатуры](#)

MOUSEINPUT

Справочные материалы

[SendInput](#)

[keybd_event](#)

[mouse_event](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция IsWindowEnabled (winuser.h)

Статья 27.08.2023

Определяет, включено ли в указанном окне ввод с помощью мыши и клавиатуры.

Синтаксис

C++

```
BOOL IsWindowEnabled(  
    [in] HWND hWnd  
);
```

Параметры

[in] hWnd

Тип: **HWND**

Дескриптор для проверяемого окна.

Возвращаемое значение

Тип: **BOOL**

Если окно включено, возвращаемое значение не равно нулю.

Если окно не включено, возвращаемое значение равно нулю.

Комментарии

Дочернее окно получает входные данные только в том случае, если оно одновременно включено и отображается.

Требования

Минимальная версия
клиента

Windows 2000 Professional [только классические приложения]

Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-window-l1-1-4 (представлено в Windows 10 версии 10.0.14393)

См. также раздел

[Основные понятия](#)

[EnableWindow](#)

[IsWindowVisible](#)

[Ввод с клавиатуры](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

функция keybd_event (winuser.h)

Статья 27.08.2023

Синтезирует нажатие клавиши. Система может использовать такой синтезированный нажатие клавиши для создания [WM_KEYUP](#) или [WM_KEYDOWN](#) сообщения. Обработчик прерываний драйвера клавиатуры вызывает функцию `keybd_event`.

Примечание Эта функция заменена. Вместо этого используйте [SendInput](#).

Синтаксис

C++

```
void keybd_event(
    [in] BYTE      bVk,
    [in] BYTE      bScan,
    [in] DWORD     dwFlags,
    [in] ULONG_PTR dwExtraInfo
);
```

Параметры

[in] bVk

Тип: **BYTE**

Код виртуального ключа. Код должен быть значением в диапазоне от 1 до 254. Полный список см. в разделе [Коды виртуальных ключей](#).

[in] bScan

Тип: **BYTE**

Код аппаратного сканирования ключа.

[in] dwFlags

Тип: **DWORD**

Управляет различными аспектами работы функции. Этот параметр может быть одним или несколькими из следующих значений.

Значение	Значение
KEYEVENTF_EXTENDEDKEY 0x0001	Если этот параметр указан, коду сканирования предшествовал байт префикса со значением 0xE0 (224).
KEYEVENTF_KEYUP 0x0002	Если этот параметр указан, ключ освобождается. Если этот параметр не указан, ключ будет удручен.

[in] dwExtraInfo

Тип: **ULONG_PTR**

Дополнительное значение, связанное с росчерком клавиши.

Возвращаемое значение

None

Remarks

Приложение может имитировать нажатие клавиши PRINTSCRN, чтобы получить snapshot экрана и сохранить его в буфер обмена. Для этого вызовите **keybd_event** с параметром *bVk*, равным **VK_SNAPSHOT**.

Примеры

Следующий пример программы переключает свет NUM LOCK с помощью **keybd_event** с виртуальной клавишей **VK_NUMLOCK**. Принимает логическое значение, указывающее, должен ли свет быть выключен (**FALSE**) или включен (**TRUE**). Тот же метод можно использовать для клавиш CAPS LOCK (**VK_CAPITAL**) и SCROLL LOCK (**VK_SCROLL**).

```
#include <windows.h>

void SetNumLock( BOOL bState )
{
    BYTE keyState[256];
```

```

GetKeyboardState((LPBYTE)&keyState);
if( (bState && !(keyState[VK_NUMLOCK] & 1)) ||
    (!bState && (keyState[VK_NUMLOCK] & 1)) )
{
// Simulate a key press
keybd_event( VK_NUMLOCK,
              0x45,
              KEYEVENTF_EXTENDEDKEY | 0,
              0 );

// Simulate a key release
keybd_event( VK_NUMLOCK,
              0x45,
              KEYEVENTF_EXTENDEDKEY | KEYEVENTF_KEYUP,
              0 );
}

void main()
{
    SetNumLock( TRUE );
}

```

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyState](#)
- [GetKeyboardState](#)

- GetSystemMetrics
 - MapVirtualKey
 - SetKeyboardState
 - Ввод с клавиатуры
-

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура KEYBDINPUT (winuser.h)

Статья 20.05.2023

Содержит сведения о событии имитации клавиатуры.

Синтаксис

C++

```
typedef struct tagKEYBDINPUT {
    WORD      wVk;
    WORD      wScan;
    DWORD     dwFlags;
    DWORD     time;
    ULONG_PTR dwExtraInfo;
} KEYBDINPUT, *PKEYBDINPUT, *LPKEYBDINPUT;
```

Члены

wVk

Тип: WORD

Код виртуального ключа. Код должен быть значением в диапазоне от 1 до 254.

Если элемент dwFlags указывает KEYEVENTF_UNICODE, wVk должен иметь значение 0.

wScan

Тип: WORD

Код аппаратного сканирования ключа. Если dwFlags указывает KEYEVENTF_UNICODE, wScan задает символ Юникода, который должен быть отправлен в приложение переднего плана.

dwFlags

Тип: DWORD

Задает различные аспекты нажатия клавиши. Этот элемент может быть определенным сочетанием следующих значений.

Значение

Значение

KEYEVENTF_EXTENDEDKEY 0x0001	Если указано, код проверки wScan состоит из последовательности из двух байтов, где первый байт имеет значение 0xE0. Дополнительные сведения см. в разделе Флаг расширенного ключа .
KEYEVENTF_KEYUP 0x0002	Если этот параметр указан, ключ освобождается. Если не указано, нажимается клавиша.
KEYEVENTF_SCANCODE 0x0008	Если этот параметр указан, wScan определяет ключ, и wVk игнорируется.
KEYEVENTF_UNICODE 0x0004	Если этот параметр указан, система синтезирует VK_PACKET нажатие клавиши. Параметр wVk должен быть равен нулю. Этот флаг можно объединить только с флагом KEYEVENTF_KEYUP . Дополнительные сведения см. в разделе «Примечания».

time

Тип: **DWORD**

Метка времени для события в миллисекундах. Если этот параметр равен нулю, система предоставит собственную метку времени.

dwExtraInfo

Тип: **ULONG_PTR**

Дополнительное значение, связанное с нажатием клавиши. Для получения этих сведений используйте функцию [GetMessageExtraInfo](#).

Комментарии

INPUT_KEYBOARD поддерживает неключевые методы ввода, такие как распознавание рукописного ввода или распознавание голоса, как если бы это был текстовый ввод с помощью флага **KEYEVENTF_UNICODE**. Если указано **KEYEVENTF_UNICODE**, **SendInput** отправляет **WM_KEYDOWN** или **WM_KEYUP** сообщение в очередь сообщений потока переднего плана с **wParam**, равным **VK_PACKET**. Когда **GetMessage** или **PeekMessage** получает это сообщение, передав его в **TranslateMessage**, публикует **WM_CHAR** сообщение с символом Юникода, изначально заданным **wScan**. Этот символ Юникода будет автоматически преобразован в соответствующее значение ANSI, если он будет размещен в окне ANSI.

Установите флаг **KEYEVENTF_SCANCODE**, чтобы определить ввод с помощью клавиатуры с точки зрения кода сканирования. Это полезно для имитации физического нажатия клавиши независимо от того, какая клавиатура используется в настоящее время. Вы также можете передать флаг **KEYEVENTF_EXTENDEDKEY**, если код сканирования является расширенным ключом. Значение виртуальной клавиши может изменяться в зависимости от текущей раскладки клавиатуры или от того, какие другие клавиши были нажаты, но код сканирования всегда будет одинаковым.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

- [GetMessageExtraInfo](#)
- [INPUT](#)
- [SendInput](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура LASTINPUTINFO (winuser.h)

Статья 28.08.2023

Содержит время последнего ввода.

Синтаксис

C++

```
typedef struct tagLASTINPUTINFO {
    UINT cbSize;
    DWORD dwTime;
} LASTINPUTINFO, *PLASTINPUTINFO;
```

Члены

`cbSize`

Тип: `UINT`

Размер структуры в байтах. Для этого элемента должно быть задано значение `sizeof(LASTINPUTINFO)`.

`dwTime`

Тип: `DWORD`

Число тактов при получении последнего входного события.

Комментарии

Эта функция полезна для обнаружения бездействия входных данных.

Дополнительные сведения о количестве тактов см. в разделе [GetTickCount](#).

Требования

Минимальная версия
клиента

Windows 2000 Professional [только классические
приложения]

Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[GetLastInputInfo](#)

[GetTickCount](#)

[Ввод с клавиатуры](#)

Справочные материалы

Обратная связь

Были ли сведения на этой странице полезными?

 Да  Нет

[Получить справку в Microsoft Q&A](#)

Функция LoadKeyboardLayoutA (winuser.h)

Статья 01.07.2023

Загружает в систему новый идентификатор языкового стандарта ввода (прежнее название — раскладка клавиатуры).

До Windows 8: Одновременно можно загрузить несколько идентификаторов входных языковых стандартов, но за один процесс активен только один. Загрузка нескольких идентификаторов входных языковых стандартов позволяет быстро переключаться между ними.

Начиная с Windows 8: Идентификатор входного языкового стандарта загружается для всей системы. Эта функция не действует, если текущий процесс не владеет окном с фокусом клавиатуры.

Синтаксис

C++

```
HKL LoadKeyboardLayoutA(
    [in] LPCSTR pwszKLID,
    [in] UINT    Flags
);
```

Параметры

[in] pwszKLID

Тип: LPCTSTR

Имя идентификатора входного языкового стандарта для загрузки. Это имя представляет собой строку, состоящую из шестнадцатеричного значения [идентификатора языка](#) (низкое слово) и идентификатора устройства (высокое слово). Например, английский язык США имеет идентификатор языка 0x0409, поэтому основной макет для английского языка США называется "00000409". Варианты американского английского макета (например, макет Дворжака) называются "00010409", "00020409" и т. д.

Список макетов ввода, которые предоставляются вместе с Windows, см. в разделе [Идентификаторы клавиатуры и редакторы методов ввода для Windows](#).

[in] Flags

Тип: **UINT**

Указывает способ загрузки идентификатора входного языкового стандарта. Этот параметр может иметь одно или несколько из следующих значений.

Значение	Значение
KLF_ACTIVATE 0x00000001	<p>До Windows 8: Если указанный идентификатор входного языкового стандарта еще не загружен, функция загружает и активирует идентификатор входного языкового стандарта для текущего потока.</p> <p>Начиная с Windows 8: Если указанный идентификатор входного языкового стандарта еще не загружен, функция загружает и активирует идентификатор входного языкового стандарта для системы.</p>
KLF_NOTELLSHELL 0x00000080	<p>До Windows 8: Запрещает процедуре перехватчика ShellProc получать код перехватчика HSHELL_LANGUAGE при загрузке нового идентификатора входного языкового стандарта. Это значение обычно используется, когда приложение загружает несколько входных идентификаторов языкового стандарта один за другим. Применение этого значения ко всем идентификаторам языкового стандарта, кроме последнего, задерживает обработку оболочки до тех пор, пока не будут добавлены все идентификаторы входного языкового стандарта.</p> <p>Начиная с Windows 8: В этом сценарии последний идентификатор входного языкового стандарта задается для всей системы.</p>
KLF_REORDER 0x00000008	<p>До Windows 8: Перемещает указанный идентификатор входного языкового стандарта в заголовок списка идентификаторов входного языкового стандарта, делая его активным идентификатором языкового стандарта для текущего потока. Это значение переупорядочение списка идентификаторов входного языкового стандарта, даже если KLF_ACTIVATE не указан.</p> <p>Начиная с Windows 8: Перемещает указанный идентификатор входного языкового стандарта в заголовок списка идентификаторов входного</p>

языкового стандарта, делая его активным идентификатором языкового стандарта для системы. Это значение переупорядочение списка идентификаторов входного языкового стандарта, даже если **KLF_ACTIVATE** не указан.

KLF_REPLACELANG 0x00000010	Если новый идентификатор языкового стандарта ввода имеет тот же идентификатор языка, что и идентификатор текущего входного языкового стандарта, новый идентификатор языкового стандарта для ввода заменяет текущий в качестве идентификатора входного языкового стандарта для этого языка. Если это значение не указано и идентификаторы входного языкового стандарта имеют одинаковые идентификаторы языка, текущий идентификатор языкового стандарта не заменяется и функция возвращает значение NULL .
KLF_SUBSTITUTE_OK 0x00000002	Заменяет указанный идентификатор входного языкового стандарта другим языковым стандартом, предпочтаемым пользователем. Система начинается с установленного флага, и рекомендуется, чтобы приложение всегда использовало этот флаг. Подстановка происходит только в том случае , если раздел реестра HKEY_CURRENT_USER\Keyboard Layout\Substitutes явно определяет языковой стандарт подстановки. Например, если ключ содержит имя значения "00000409" со значением "00010409", загрузка макета США ("00000409") приведет к загрузке макета States-Dvorak ("00010409"). Система использует KLF_SUBSTITUTE_OK при загрузке, и рекомендуется, чтобы все приложения использовали это значение при загрузке идентификаторов входных языковых стандартов, чтобы убедиться, что выбраны предпочтения пользователя.
KLF_SETFORPROCESS 0x00000100	До Windows 8: Этот флаг действителен только для KLF_ACTIVATE . Активирует указанный идентификатор языкового стандарта для всего процесса и отправляет сообщение WM_INPUTLANGCHANGE в окно Фокус текущего потока или Активно. Как правило, LoadKeyboardLayout активирует идентификатор входного языкового стандарта только для текущего потока. Начиная с Windows 8: Этот флаг не используется. LoadKeyboardLayout всегда активирует идентификатор языкового стандарта для всей системы, если текущий процесс владеет окном с фокусом клавиатуры.

KLF_UNLOADPREVIOUS

Этот флаг не поддерживается. Вместо этого используйте функцию [UnloadKeyboardLayout](#).

Возвращаемое значение

Тип: HKL

Если функция выполняется успешно, возвращаемое значение — это идентификатор входного языкового стандарта, соответствующий имени, указанному в *pwszKLID*. Если соответствующий языковой стандарт недоступен, возвращаемое значение является языком системы по умолчанию.

Если функция завершается сбоем, возвращается значение NULL. Это может произойти, если библиотека макета загружена из каталога приложения.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Приложение может и обычно загружает идентификатор входного языкового стандарта по умолчанию или IME для языка и может сделать это, указав только строковую версию идентификатора языка. Если приложение хочет загрузить определенный языковой стандарт или IME, оно должно прочитать реестр, чтобы определить конкретный идентификатор входного языкового стандарта для передачи в [LoadKeyboardLayout](#). В этом случае запрос на активацию идентификатора входного языкового стандарта по умолчанию для языкового стандарта активирует первый соответствующий. Определенный IME следует активировать с помощью явного входного идентификатора языкового стандарта, возвращаемого из [GetKeyboardLayout](#) или [LoadKeyboardLayout](#).

До Windows 8: Эта функция влияет только на макет для текущего процесса или потока.

Начиная с Windows 8: Эта функция влияет на макет всей системы.

ⓘ Примечание

Заголовок winuser.h определяет LoadKeyboardLayout в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОДа. Использование псевдонима, не зависящий от кодирования, с кодом, который не является нейтральным для кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или времени выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[ActivateKeyboardLayout](#)

[Основные понятия](#)

[GetKeyboardLayoutName](#)

[Ввод с клавиатуры](#)

[MAKELANGID](#)

[Другие ресурсы](#)

[Справочные материалы](#)

[UnloadKeyboardLayout](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция LoadKeyboardLayoutW (winuser.h)

Статья 11.03.2023

Загружает в систему новый идентификатор языкового стандарта ввода (прежнее название — раскладка клавиатуры).

До Windows 8. Одновременно можно загрузить несколько идентификаторов входных языковых стандартов, но за раз активируется только один процесс. Загрузка нескольких идентификаторов входных языковых стандартов позволяет быстро переключаться между ними.

Начиная с Windows 8: идентификатор входного языкового стандарта загружается для всей системы. Эта функция не действует, если текущий процесс не владеет окном с фокусом клавиатуры.

Синтаксис

C++

```
HKL LoadKeyboardLayoutW(
    [in] LPCWSTR pwszKLID,
    [in] UINT     Flags
);
```

Параметры

[in] pwszKLID

Тип: LPCTSTR

Имя идентификатора входного языкового стандарта для загрузки. Это имя представляет собой строку, состоящую из шестнадцатеричного значения [идентификатора языка](#) (низкое слово) и идентификатора устройства (высокое слово). Например, английский язык США имеет идентификатор языка 0x0409, поэтому основной макет для английского языка США называется "00000409". Варианты американского английского макета (например, макет Дворжака) называются "00010409", "00020409" и т. д.

Список макетов ввода, которые предоставляются вместе с Windows, см. в разделе [Идентификаторы клавиатуры и редакторы методов ввода для Windows](#).

[in] Flags

Тип: **UINT**

Указывает способ загрузки идентификатора входного языкового стандарта. Этот параметр может принимать одно из указанных ниже значений.

Значение	Значение
KLF_ACTIVATE 0x00000001	<p>До Windows 8. Если указанный идентификатор входного языкового стандарта еще не загружен, функция загружает и активирует идентификатор входного языкового стандарта для текущего потока.</p> <p>Начиная с Windows 8: если указанный идентификатор входного языкового стандарта еще не загружен, функция загружает и активирует идентификатор входного языкового стандарта для системы.</p>
KLF_NOTELLSHELL 0x00000080	<p>До Windows 8. Запрещает процедуре перехватчика ShellProc получать код перехватчика HSHELL_LANGUAGE при загрузке нового идентификатора входного языкового стандарта. Это значение обычно используется, когда приложение загружает несколько входных идентификаторов языкового стандарта один за другим. Применение этого значения ко всем идентификаторам языкового стандарта, кроме последнего, задерживает обработку оболочки до тех пор, пока не будут добавлены все входные идентификаторы языкового стандарта.</p> <p>Начиная с Windows 8. В этом сценарии последний идентификатор входного языкового стандарта устанавливается для всей системы.</p>
KLF_REORDER 0x00000008	<p>До Windows 8. Перемещает указанный идентификатор входного языкового стандарта в начало списка идентификаторов входного языкового стандарта, делая этот идентификатор активным идентификатором языкового стандарта для текущего потока. Это значение переупорядочение списка идентификаторов входного языкового стандарта, даже если KLF_ACTIVATE не указан.</p> <p>Начиная с Windows 8. Перемещает указанный идентификатор входного языкового стандарта в начало списка идентификаторов входного языкового</p>

стандарта, делая этот идентификатор активным идентификатором языкового стандарта для системы. Это значение переупорядочение списка идентификаторов входного языкового стандарта, даже если **KLF_ACTIVATE** не указан.

KLF_REPLACELANG 0x00000010	Если новый идентификатор языкового стандарта ввода имеет тот же идентификатор языка, что и идентификатор текущего входного языкового стандарта, новый идентификатор языкового стандарта для ввода заменяет текущий в качестве идентификатора входного языкового стандарта для этого языка. Если это значение не указано и идентификаторы входного языкового стандарта имеют одинаковые идентификаторы языка, текущий идентификатор языкового стандарта не заменяется и функция возвращает значение NULL .
KLF_SUBSTITUTE_OK 0x00000002	Заменяет указанный идентификатор входного языкового стандарта другим языковым стандартом, предпочтаемым пользователем. Система начинается с установленного флага, и рекомендуется, чтобы приложение всегда использовало этот флаг. Подстановка происходит только в том случае , если раздел реестра HKEY_CURRENT_USER\Keyboard Layout\Substitutes явно определяет языковой стандарт подстановки. Например, если ключ содержит имя значения "00000409" со значением "00010409", загрузка макета США ("00000409") приведет к загрузке макета States-Dvorak ("00010409"). Система использует KLF_SUBSTITUTE_OK при загрузке, и рекомендуется, чтобы все приложения использовали это значение при загрузке идентификаторов входных языковых стандартов, чтобы убедиться, что выбраны предпочтения пользователя.
KLF_SETFORPROCESS 0x00000100	<p>До Windows 8: этот флаг действителен только для KLF_ACTIVATE. Активирует указанный идентификатор языкового стандарта для всего процесса и отправляет сообщение WM_INPUTLANGCHANGE в окно Фокус текущего потока или Активно. Как правило, LoadKeyboardLayout активирует идентификатор входного языкового стандарта только для текущего потока.</p> <p>Начиная с Windows 8: этот флаг не используется. LoadKeyboardLayout всегда активирует идентификатор языкового стандарта для всей системы, если текущий процесс владеет окном с фокусом клавиатуры.</p>

KLF_UNLOADPREVIOUS

Этот флаг не поддерживается. Вместо этого используйте функцию [UnloadKeyboardLayout](#).

Возвращаемое значение

Тип: HKL

Если функция выполняется успешно, возвращаемое значение — это идентификатор входного языкового стандарта, соответствующий имени, указанному в *pwszKLID*. Если соответствующий языковой стандарт недоступен, возвращаемое значение является языком по умолчанию в системе.

Если функция завершается сбоем, возвращается значение NULL. Это может произойти, если библиотека макета загружена из каталога приложения.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Приложение может и обычно загружает идентификатор входного языкового стандарта по умолчанию или IME для языка и может сделать это, указав только строковую версию идентификатора языка. Если приложение хочет загрузить определенный языковой стандарт или IME, оно должно прочитать реестр, чтобы определить конкретный идентификатор входного языкового стандарта для передачи в [LoadKeyboardLayout](#). В этом случае запрос на активацию идентификатора входного языкового стандарта по умолчанию для языкового стандарта активирует первый соответствующий. Определенный IME следует активировать с помощью явного входного идентификатора языкового стандарта, возвращаемого из [GetKeyboardLayout](#) или [LoadKeyboardLayout](#).

До Windows 8. Эта функция влияет только на макет текущего процесса или потока.

Начиная с Windows 8: эта функция влияет на макет всей системы.

ⓘ Примечание

Заголовок `winuser.h` определяет `LoadKeyboardLayout` в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на

основе определения константы препроцессора ЮНИКОД. Использование псевдонима, не зависящий от кодирования, с кодом, который не является нейтральным для кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или времени выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[ActivateKeyboardLayout](#)

Основные понятия

[GetKeyboardLayoutName](#)

[Ввод с клавиатуры](#)

[MAKELANGID](#)

Другие ресурсы

Справочные материалы

[UnloadKeyboardLayout](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция MapVirtualKeyA (winuser.h)

Статья 27.08.2023

Преобразует (сопоставляет) код виртуального ключа в код сканирования или значение символа или преобразует код сканирования в код виртуального ключа.

Синтаксис

C++

```
UINT MapVirtualKeyA(  
    [in] UINT uCode,  
    [in] UINT uMapType  
);
```

Параметры

[in] uCode

Тип: **UINT**

[Код виртуального ключа](#) или код сканирования ключа. Способ интерпретации этого значения зависит от значения параметра *uMapType*.

Начиная с Windows Vista, высокий байт значения *uCode* может содержать 0xe0 или 0xe1 для указания расширенного кода сканирования.

[in] uMapType

Тип: **UINT**

Выполняемый перевод. Значение этого параметра зависит от значения параметра *uCode*.

Значение	Значение
MAPVK_VK_TO_VSC 0	Параметр <i>uCode</i> представляет собой код виртуального ключа и преобразуется в код сканирования. Если это код виртуального ключа, который не различает левый и правый ключи, возвращается код сканирования слева. Если перевода нет, функция возвращает значение 0.

Значение	Значение
MAPVK_VSC_TO_VK 1	Параметр <i>iCode</i> представляет собой код сканирования и преобразуется в код виртуального ключа, который не различает левый и правый ключи. Если перевода нет, функция возвращает значение 0.
MAPVK_VK_TO_CHAR 2	Параметр <i>iCode</i> представляет собой код виртуального ключа и преобразуется в неперемеченное символьное значение в слове нижнего порядка возвращаемого значения. Мертвые ключи (диакритические знаки) указываются путем задания верхнего бита возвращаемого значения. Если перевода нет, функция возвращает значение 0. См. заметки.
MAPVK_VSC_TO_VK_EX 3	Параметр <i>iCode</i> представляет собой код сканирования и преобразуется в код виртуального ключа, который различает левый и правый ключи. Если перевода нет, функция возвращает значение 0.
MAPVK_VK_TO_VSC_EX 4	Windows Vista и более поздних версий: Параметр <i>iCode</i> представляет собой код виртуального ключа и преобразуется в код сканирования. Если это код виртуального ключа, который не различает левый и правый ключи, возвращается код сканирования слева. Если код сканирования является расширенным, высокий байт значения <i>iCode</i> может содержать либо 0xe0, либо 0xe1 для указания расширенного кода сканирования. Если перевода нет, функция возвращает значение 0.

Возвращаемое значение

Тип: **UINT**

Возвращаемое значение представляет собой код сканирования, код виртуального ключа или символьное значение в зависимости от значения *iCode* и *iMapType*. Если перевода нет, возвращаемое значение равно нулю.

Комментарии

Чтобы указать дескриптор раскладки клавиатуры для перевода указанного кода, используйте функцию [MapVirtualKeyEx](#).

Приложение может использовать [MapVirtualKey](#) для преобразования кодов сканирования в константы кода виртуального ключа **VK_SHIFT**, **VK_CONTROL** и **VK_MENU** и наоборот. Эти переводы не различают левый и правый экземпляры клавиш SHIFT, CTRL или ALT.

Приложение может получить код сканирования, соответствующий левому или правому экземпляру одного из этих ключей, вызвав **MapVirtualKey** с параметром *uCode*, который имеет одну из следующих констант кода виртуального ключа:

- VK_LSHIFT
- VK_RSHIFT
- VK_LCONTROL
- VK_RCONTROL
- VK_LMENU
- VK_RMENU

Эти различающиеся слева и справа константы доступны приложению только с помощью функций [GetKeyboardState](#), [SetKeyboardState](#), [GetAsyncKeyState](#), [MapVirtualKey](#) и [MapVirtualKeyEx](#). Полный список кодов виртуальных ключей см. в разделе [Коды виртуальных ключей](#).

В режиме **MAPVK_VK_TO_CHAR** [коды виртуального ключа](#) — "A". Клавиши Z преобразуются в верхний регистр "A". Символы Z независимо от текущей раскладки клавиатуры. Если вы хотите перевести код виртуального ключа в соответствующий символ, используйте функцию [ToAscii](#).

ⓘ Примечание

Заголовок winuser.h определяет MapVirtualKey в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows

Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [GetSystemMetrics](#)
- [MapVirtualKey](#)
- [SetKeyboardState](#)
- [Ввод с клавиатуры](#)
- [Общие сведения о вводе с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция MapVirtualKeyExA (winuser.h)

Статья 15.03.2023

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. Функция преобразует коды с помощью языка ввода и идентификатора языкового стандарта.

Синтаксис

C++

```
UINT MapVirtualKeyExA(
    [in]                 UINT uCode,
    [in]                 UINT uMapType,
    [in, out, optional] HKL dwhkl
);
```

Параметры

[in] uCode

Тип: **UINT**

[Код виртуального ключа](#) или код сканирования ключа. Способ интерпретации этого значения зависит от значения параметра *uMapType*.

Начиная с **Windows Vista**, высокий байт значения *uCode* может содержать 0xe0 или 0xe1 для указания расширенного кода сканирования.

[in] uMapType

Тип: **UINT**

Выполняемый перевод. Значение этого параметра зависит от значения параметра *uCode*.

Значение

Значение

Значение	Значение
MAPVK_VK_TO_VSC 0	Параметр <i>uCode</i> представляет собой код виртуального ключа и преобразуется в код сканирования. Если это код виртуального ключа, который не различает левый и правый ключи, возвращается код сканирования слева. Если перевода нет, функция возвращает значение 0.
MAPVK_VSC_TO_VK 1	Параметр <i>uCode</i> представляет собой код сканирования и преобразуется в код виртуального ключа, который не различает левый и правый ключи. Если перевода нет, функция возвращает значение 0.
MAPVK_VK_TO_CHAR 2	Параметр <i>uCode</i> — это код виртуального ключа, который преобразуется в неперемеченное символьное значение в слове нижнего порядка возвращаемого значения. Мертвые ключи (диакритические знаки) указываются путем задания верхнего бита возвращаемого значения. Если перевода нет, функция возвращает значение 0. См. заметки.
MAPVK_VSC_TO_VK_EX 3	Параметр <i>uCode</i> представляет собой код сканирования и преобразуется в код виртуального ключа, который различает левый и правый ключи. Если перевода нет, функция возвращает значение 0.
MAPVK_VK_TO_VSC_EX 4	Windows Vista и более поздних версий: Параметр <i>uCode</i> представляет собой код виртуального ключа и преобразуется в код сканирования. Если это код виртуального ключа, который не различает левый и правый ключи, возвращается код сканирования слева. Если код сканирования является расширенным, высокий байт значения <i>uCode</i> может содержать либо 0xe0, либо 0xe1 для указания расширенного кода сканирования. Если перевода нет, функция возвращает значение 0.

[in, out, optional] dwhk1

Тип: HKL

Входной идентификатор языкового стандарта, используемый для перевода указанного кода. Этот параметр может быть любым идентификатором входного языкового стандарта, ранее возвращенным функцией [LoadKeyboardLayout](#).

Возвращаемое значение

Тип: UINT

Возвращаемое значение представляет собой код сканирования, код виртуального ключа или символьное значение в зависимости от значения *uCode* и *uMapType*.

Если перевода нет, возвращаемое значение равно нулю.

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Приложение может использовать `MapVirtualKeyEx` для преобразования кодов сканирования в константы кода виртуального ключа `VK_SHIFT`, `VK_CONTROL` и `VK_MENU` и наоборот. Эти переводы не различают левый и правый экземпляры клавиш SHIFT, CTRL или ALT.

Приложение может получить код сканирования, соответствующий левому или правому экземпляру одного из этих ключей, вызвав `MapVirtualKeyEx` с параметром *iCode*, который имеет одну из следующих констант кода виртуального ключа:

- `VK_LSHIFT`
- `VK_RSHIFT`
- `VK_LCONTROL`
- `VK_RCONTROL`
- `VK_LMENU`
- `VK_RMENU`

Эти различающиеся слева и справа константы доступны приложению только с помощью функций `GetKeyboardState`, `SetKeyboardState`, `GetAsyncKeyState`, `MapVirtualKey` и `MapVirtualKeyEx`. Полный список кодов виртуальных ключей см. в разделе [Коды виртуальных ключей](#).

В режиме `MAPVK_VK_TO_CHAR` коды виртуального ключа — "A". Клавиши Z преобразуются в верхний регистр "A". Символы Z независимо от текущей раскладки клавиатуры. Если вы хотите преобразовать код виртуального ключа в соответствующий символ, используйте функцию `ToAscii`.

ⓘ Примечание

Заголовок `winuser.h` определяет `MapVirtualKeyEx` как псевдоним, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от кодировки, может привести к несоответствиям, которые приводят к ошибкам

компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [GetSystemMetrics](#)
- [MapVirtualKey](#)
- [SetKeyboardState](#)
- [LoadKeyboardLayout](#)
- [Ввод с клавиатуры](#)
- [Общие сведения о вводе с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция MapVirtualKeyExW (winuser.h)

Статья 06.04.2023

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа. Функция преобразует коды с помощью языка ввода и идентификатора языкового стандарта.

Синтаксис

C++

```
UINT MapVirtualKeyExW(
    [in]                 UINT uCode,
    [in]                 UINT uMapType,
    [in, out, optional] HKL dwhkl
);
```

Параметры

[in] uCode

Тип: **UINT**

[Код виртуального ключа](#) или код сканирования ключа. Способ интерпретации этого значения зависит от значения параметра *uMapType*.

Начиная с **Windows Vista**, высокий байт значения *uCode* может содержать 0xe0 или 0xe1 для указания расширенного кода сканирования.

[in] uMapType

Тип: **UINT**

Выполняемый перевод. Значение этого параметра зависит от значения параметра *uCode*.

Значение

Значение

Значение	Значение
MAPVK_VK_TO_VSC 0	Параметр <i>uCode</i> представляет собой код виртуального ключа и преобразуется в код сканирования. Если это код виртуального ключа, который не различает левый и правый ключи, возвращается код сканирования слева. Если перевода нет, функция возвращает значение 0.
MAPVK_VSC_TO_VK 1	Параметр <i>uCode</i> представляет собой код сканирования и преобразуется в код виртуального ключа, который не различает левый и правый ключи. Если перевода нет, функция возвращает значение 0.
MAPVK_VK_TO_CHAR 2	Параметр <i>uCode</i> — это код виртуального ключа, который преобразуется в неперемеченное символьное значение в слове нижнего порядка возвращаемого значения. Мертвые ключи (диакритические знаки) указываются путем задания верхнего бита возвращаемого значения. Если перевода нет, функция возвращает значение 0. См. заметки.
MAPVK_VSC_TO_VK_EX 3	Параметр <i>uCode</i> представляет собой код сканирования и преобразуется в код виртуального ключа, который различает левый и правый ключи. Если перевода нет, функция возвращает значение 0.
MAPVK_VK_TO_VSC_EX 4	Windows Vista и более поздних версий: Параметр <i>uCode</i> представляет собой код виртуального ключа и преобразуется в код сканирования. Если это код виртуального ключа, который не различает левый и правый ключи, возвращается код сканирования слева. Если код сканирования является расширенным, высокий байт значения <i>uCode</i> может содержать либо 0xe0, либо 0xe1 для указания расширенного кода сканирования. Если перевода нет, функция возвращает значение 0.

[in, out, optional] dwhk1

Тип: HKL

Входной идентификатор языкового стандарта, используемый для перевода указанного кода. Этот параметр может быть любым идентификатором входного языкового стандарта, ранее возвращенным функцией [LoadKeyboardLayout](#).

Возвращаемое значение

Тип: UINT

Возвращаемое значение представляет собой код сканирования, код виртуального ключа или символьное значение в зависимости от значения *uCode* и *uMapType*.

Если перевода нет, возвращаемое значение равно нулю.

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Приложение может использовать `MapVirtualKeyEx` для преобразования кодов сканирования в константы кода виртуального ключа `VK_SHIFT`, `VK_CONTROL` и `VK_MENU` и наоборот. Эти переводы не различают левый и правый экземпляры клавиш SHIFT, CTRL или ALT.

Приложение может получить код сканирования, соответствующий левому или правому экземпляру одного из этих ключей, вызвав `MapVirtualKeyEx` с параметром *iCode*, который имеет одну из следующих констант кода виртуального ключа:

- `VK_LSHIFT`
- `VK_RSHIFT`
- `VK_LCONTROL`
- `VK_RCONTROL`
- `VK_LMENU`
- `VK_RMENU`

Эти различающиеся слева и справа константы доступны приложению только с помощью функций `GetKeyboardState`, `SetKeyboardState`, `GetAsyncKeyState`, `MapVirtualKey` и `MapVirtualKeyEx`. Полный список кодов виртуальных ключей см. в разделе [Коды виртуальных ключей](#).

В режиме `MAPVK_VK_TO_CHAR` коды виртуального ключа — "A". Клавиши Z преобразуются в верхний регистр "A". Символы Z независимо от текущей раскладки клавиатуры. Если вы хотите перевести код виртуального ключа в соответствующий символ, используйте функцию [ToUnicode](#).

ⓘ Примечание

Заголовок `winuser.h` определяет `MapVirtualKeyEx` как псевдоним, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от кодировки, может привести к несоответствиям, которые приводят к ошибкам

компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [GetSystemMetrics](#)
- [MapVirtualKey](#)
- [SetKeyboardState](#)
- [LoadKeyboardLayout](#)
- [Ввод с клавиатуры](#)
- [Общие сведения о вводе с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция MapVirtualKeyW (winuser.h)

Статья 15.03.2023

Преобразует (сопоставляет) код виртуального ключа в код сканирования или символьное значение или преобразует код сканирования в код виртуального ключа.

Синтаксис

C++

```
UINT MapVirtualKeyW(
    [in] UINT uCode,
    [in] UINT uMapType
);
```

Параметры

[in] uCode

Тип: **UINT**

[Код виртуального ключа](#) или код сканирования ключа. Способ интерпретации этого значения зависит от значения параметра *uMapType*.

Начиная с **Windows Vista**, высокий байт значения *uCode* может содержать 0xe0 или 0xe1 для указания расширенного кода сканирования.

[in] uMapType

Тип: **UINT**

Выполняемый перевод. Значение этого параметра зависит от значения параметра *uCode*.

Значение	Значение
MAPVK_VK_TO_VSC 0	Параметр <i>uCode</i> представляет собой код виртуального ключа и преобразуется в код сканирования. Если это код виртуального ключа, который не различает левый и правый ключи, возвращается код сканирования слева. Если перевода нет, функция возвращает значение 0.

Значение	Значение
MAPVK_VSC_TO_VK 1	Параметр <i>iCode</i> представляет собой код сканирования и преобразуется в код виртуального ключа, который не различает левый и правый ключи. Если перевода нет, функция возвращает значение 0.
MAPVK_VK_TO_CHAR 2	Параметр <i>iCode</i> — это код виртуального ключа, который преобразуется в неперемеченное символьное значение в слове нижнего порядка возвращаемого значения. Мертвые ключи (диакритические знаки) указываются путем задания верхнего бита возвращаемого значения. Если перевода нет, функция возвращает значение 0. См. заметки.
MAPVK_VSC_TO_VK_EX 3	Параметр <i>iCode</i> представляет собой код сканирования и преобразуется в код виртуального ключа, который различает левый и правый ключи. Если перевода нет, функция возвращает значение 0.
MAPVK_VK_TO_VSC_EX 4	Windows Vista и более поздних версий: Параметр <i>iCode</i> представляет собой код виртуального ключа и преобразуется в код сканирования. Если это код виртуального ключа, который не различает левый и правый ключи, возвращается код сканирования слева. Если код сканирования является расширенным, высокий байт значения <i>iCode</i> может содержать либо 0xe0, либо 0xe1 для указания расширенного кода сканирования. Если перевода нет, функция возвращает значение 0.

Возвращаемое значение

Тип: **UINT**

Возвращаемое значение представляет собой код сканирования, код виртуального ключа или символьное значение в зависимости от значения *iCode* и *iMapType*. Если перевода нет, возвращаемое значение равно нулю.

Комментарии

Чтобы указать дескриптор раскладки клавиатуры для перевода указанного кода, используйте функцию [MapVirtualKeyEx](#).

Приложение может использовать [MapVirtualKey](#) для преобразования кодов сканирования в константы кода виртуального ключа **VK_SHIFT**, **VK_CONTROL** и **VK_MENU** и наоборот. Эти переводы не различают левый и правый экземпляры клавиш SHIFT, CTRL или ALT.

Приложение может получить код сканирования, соответствующий левому или правому экземпляру одного из этих ключей, вызвав **MapVirtualKey** с параметром *uCode*, который имеет одну из следующих констант кода виртуального ключа:

- VK_LSHIFT
- VK_RSHIFT
- VK_LCONTROL
- VK_RCONTROL
- VK_LMENU
- VK_RMENU

Эти различающиеся слева и справа константы доступны приложению только с помощью функций [GetKeyboardState](#), [SetKeyboardState](#), [GetAsyncKeyState](#), [MapVirtualKey](#) и [MapVirtualKeyEx](#). Полный список кодов виртуальных ключей см. в разделе [Коды виртуальных ключей](#).

В режиме **MAPVK_VK_TO_CHAR** [коды виртуального ключа](#) — "A". Клавиши Z преобразуются в верхний регистр "A". Символы Z независимо от текущей раскладки клавиатуры. Если вы хотите перевести код виртуального ключа в соответствующий символ, используйте функцию [ToUnicode](#).

ⓘ Примечание

Заголовок winuser.h определяет MapVirtualKey в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows

Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [GetSystemMetrics](#)
- [MapVirtualKey](#)
- [MapVirtualKeyEx](#)
- [SetKeyboardState](#)
- [Ввод с клавиатуры](#)
- [Общие сведения о вводе с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

функция mouse_event (winuser.h)

Статья 27.08.2023

Функция **mouse_event** синтезирует движения мыши и нажатия кнопки.

Примечание Эта функция заменена. Вместо этого используйте **SendInput**.

Синтаксис

C++

```
void mouse_event(
    [in] DWORD      dwFlags,
    [in] DWORD      dx,
    [in] DWORD      dy,
    [in] DWORD      dwData,
    [in] ULONG_PTR  dwExtraInfo
);
```

Параметры

[in] dwFlags

Тип: **DWORD**

Управляет различными аспектами движения мыши и нажатием кнопки. Этот параметр может быть определенным сочетанием следующих значений.

Значение	Значение
MOUSEEVENTF_ABSOLUTE 0x8000	Параметры <i>dx</i> и <i>dy</i> содержат нормализованные абсолютные координаты. Если значение не задано, эти параметры содержат относительные данные: изменение позиции с момента последней сообщаемой позиции. Этот флаг может быть установлен или не установлен, независимо от типа мыши или устройства, похожего на мышь, если таковой имеется, подключен к системе. Дополнительные сведения об относительном перемещении мыши см. в следующем разделе Примечаний.

MOUSEEVENTF_LEFTDOWN 0x0002	Левая кнопка не работает.
MOUSEEVENTF_LEFTUP 0x0004	Кнопка слева находится вверх.
MOUSEEVENTF_MIDDLEDOWN 0x0020	Средняя кнопка вниз.
MOUSEEVENTF_MIDDLEUP 0x0040	Средняя кнопка вверх.
MOUSEEVENTF_MOVE 0x0001	Произошло перемещение.
MOUSEEVENTF_RIGHTDOWN 0x0008	Кнопка справа вниз.
MOUSEEVENTF_RIGHTUP 0x0010	Кнопка справа находится вверх.
MOUSEEVENTF_WHEEL 0x0800	Колесо было перемещено, если у мыши есть колесо. Объем перемещения указывается в <i>dwData</i>
MOUSEEVENTF_XDOWN 0x0080	Нажата кнопка X.
MOUSEEVENTF_XUP 0x0100	Кнопка X была отпущена.
MOUSEEVENTF_WHEEL 0x0800	Кнопка колесика повернется.
MOUSEEVENTF_HWHEEL 0x01000	Кнопка колесика наклонена.

Значения, указывающие состояние кнопки мыши, задаются для указания изменений состояния, а не текущих условий. Например, если левая кнопка мыши нажата и удерживается, **MOUSEEVENTF_LEFTDOWN** устанавливается при первом нажатии левой кнопки, но не для последующих движений. Аналогичным образом **MOUSEEVENTF_LEFTUP** устанавливается только при первом освобождении кнопки.

В параметре *dwFlags* нельзя указать одновременно как **MOUSEEVENTF_WHEEL**, так и **MOUSEEVENTF_XDOWN** или **MOUSEEVENTF_XUP**, так как для обоих из них требуется использовать поле *dwData*.

[in] dx

Тип: DWORD

Абсолютное положение мыши по оси X или объем ее движения с момента создания последнего события мыши в зависимости от настройки **MOUSEEVENTF_ABSOLUTE**. Абсолютные данные указываются в качестве фактической координаты X мыши; относительные данные указываются как количество перемещенных микки. *Микки* — это количество, которое мышь должна переместить, чтобы сообщить о перемещении.

[in] dy

Тип: DWORD

Абсолютное положение мыши по оси Y или объем ее движения с момента создания последнего события мыши в зависимости от настройки **MOUSEEVENTF_ABSOLUTE**. Абсолютные данные указываются в качестве фактической координаты мыши по оси Y; относительные данные указываются как количество перемещенных микки.

[in] dwData

Тип: DWORD

Если *dwFlags* содержит **MOUSEEVENTF_WHEEL**, *dwData* указывает объем перемещения колесика. Положительное значение указывает, что колесико было повернуто вперед, от пользователя; отрицательное значение указывает, что колесико было повернуто назад к пользователю. Один щелчок колесиком определяется как **WHEEL_DELTA**, то есть 120.

Если *dwFlags* содержит **MOUSEEVENTF_HWHEEL**, *dwData* указывает объем перемещения колесика. Положительное значение указывает, что колесико было наклонено вправо; отрицательное значение указывает, что колесико было наклонено влево.

Если *dwFlags* содержит **MOUSEEVENTF_XDOWN** или **MOUSEEVENTF_XUP**, *dwData* указывает, какие кнопки X были нажаты или отпущены. Это значение может быть любым сочетанием следующих флагов.

Если параметр *dwFlags* не является **MOUSEEVENTF_WHEEL**, **MOUSEEVENTF_XDOWN** или **MOUSEEVENTF_XUP**, значение *dwData* должно быть равно нулю.

Значение	Значение
----------	----------

XBUTTON1 0x0001	Задает значение, если первая кнопка X была нажата или отпущена.
XBUTTON2 0x0002	Задает значение, если вторая кнопка X была нажата или отпущена.

[in] dwExtraInfo

Тип: **ULONG_PTR**

Дополнительное значение, связанное с событием мыши. Приложение вызывает [GetMessageExtraInfo](#), чтобы получить эти дополнительные сведения.

Возвращаемое значение

None

Remarks

Если мышь переместилась, на что указывает **MOUSEEVENTF_MOVE** устанавливается, *dx* и *dy* удерживайте сведения об этом движении. Сведения указываются как абсолютные или относительные целочисленные значения.

Если указано **MOUSEEVENTF_ABSOLUTE** значение, *dx* и *dy* содержат нормализованные абсолютные координаты в диапазоне от 0 до 65 535. Процедура события сопоставляет эти координаты с поверхностью дисплея. Координаты (0,0) сопоставляется с левым верхним углом поверхности дисплея, (65535 65535) сопоставляется с правым нижним углом.

Если значение **MOUSEEVENTF_ABSOLUTE** не указано, *dx* и *dy* указывают относительные движения с момента создания последнего события мыши (последняя указанная позиция). Положительные значения означают перемещение мыши вправо (или вниз); Отрицательные значения означают перемещение мыши влево (или вверх).

Относительное движение мыши зависит от параметров скорости и уровня ускорения мыши. Пользователь задает эти значения с помощью приложения Mouse в панель управления. Приложение получает и задает эти значения с помощью функции [SystemParametersInfo](#).

Система применяет два теста к указанному относительному движению мыши при применении ускорения. Если указанное расстояние по оси X или Y больше первого порогового значения мыши, а уровень ускорения мыши не равен нулю,

операционная система удваивает расстояние. Если указанное расстояние по оси X или Y больше второго порогового значения мыши, а уровень ускорения мыши равен двум, операционная система удваивает расстояние, полученное в результате применения первого порогового теста. Таким образом, операционная система может умножить относительно заданное движение мыши по оси X или Y до четырех раз.

После применения ускорения система масштабирует итоговое значение на нужную скорость мыши. Скорость мыши может варьироваться от 1 (самая медленная) до 20 (самая быстрая) и представляет, сколько перемещается указатель в зависимости от расстояния, которое перемещает мышь. Значение по умолчанию — 10, что не приводит к дополнительным изменениям движения мыши.

Функция `mouse_event` используется для синтеза событий мыши приложениями, которым это необходимо сделать. Он также используется приложениями, которым требуется получить от мыши больше сведений, чем ее положение и состояние кнопки. Например, если производитель планшета хочет передать информацию на основе пера в свои приложения, он может написать библиотеку DLL, которая напрямую взаимодействует с оборудованием планшета, получает дополнительные сведения и сохраняет их в очереди. Затем библиотека DLL вызывает `mouse_event` со стандартной кнопкой и данными о положении x/y, а также в параметре `dwExtraInfo` указателем или индексом на дополнительную информацию в очереди. Когда приложению требуются дополнительные сведения, оно вызывает библиотеку DLL с указателем или индексом, хранящимся в `dwExtraInfo`, и библиотека DLL возвращает дополнительные сведения.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

Основные понятия

[GetMessageExtraInfo](#)

[Ввод с помощью мыши](#)

[Другие ресурсы](#)

[Справочные материалы](#)

[SystemParametersInfo](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура MOUSEINPUT (winuser.h)

Статья 28.08.2023

Содержит сведения о имитированном событии мыши.

Синтаксис

C++

```
typedef struct tagMOUSEINPUT {
    LONG      dx;
    LONG      dy;
    DWORD     mouseData;
    DWORD     dwFlags;
    DWORD     time;
    ULONG_PTR dwExtraInfo;
} MOUSEINPUT, *PMOUSEINPUT, *LPMOUSEINPUT;
```

Члены

dx

Тип: **LONG**

Абсолютное положение мыши или объем движения с момента создания последнего события мыши в зависимости от значения элемента **dwFlags**.
Абсолютные данные указываются как координата X мыши; относительные данные указываются как количество перемещенных пикселей.

dy

Тип: **LONG**

Абсолютное положение мыши или объем движения с момента создания последнего события мыши в зависимости от значения элемента **dwFlags**.
Абсолютные данные указываются как координата Y мыши; относительные данные указываются как количество перемещенных пикселей.

mouseData

Тип: **DWORD**

Если **dwFlags** содержит **MOUSEEVENTF_WHEEL**, **mouseData** указывает величину перемещения колесика. Положительное значение указывает, что колесо повернулось вперед, от пользователя; отрицательное значение указывает, что колесо повернулось назад к пользователю. Один щелчок колесиком определяется как **WHEEL_DELTA**, то есть 120.

Windows Vista: если **dwFlags** содержит **MOUSEEVENTF_HWHEEL**, то **dwData** указывает величину перемещения колесика. Положительное значение указывает, что колесо повернулось вправо; Отрицательное значение указывает, что колесо было повернуто влево. Один щелчок колесиком определяется как **WHEEL_DELTA**, то есть 120.

Если **dwFlags** не содержит **MOUSEEVENTF_WHEEL**, **MOUSEEVENTF_XDOWN** или **MOUSEEVENTF_XUP**, значение **mouseData** должно быть равно нулю.

Если **dwFlags** содержит **MOUSEEVENTF_XDOWN** или **MOUSEEVENTF_XUP**, **mouseData** указывает, какие кнопки X были нажаты или отпущены. Это значение может быть любым сочетанием следующих флагов.

Значение	Значение
XBUTTON1 0x0001	Задает значение, если первая кнопка X нажата или отпущена.
XBUTTON2 0x0002	Задает значение , если вторая кнопка X нажата или отпущена.

dwFlags

Тип: **DWORD**

Набор битовых флагов, указывающих различные аспекты движения мыши и нажатий кнопок. Биты в этом элементе могут быть любым разумным сочетанием следующих значений.

Битовые флаги, указывающие состояние кнопки мыши, задаются для указания изменений в состоянии, а не текущих условий. Например, если левая кнопка мыши нажата и удерживается, **MOUSEEVENTF_LEFTDOWN** устанавливается при первом нажатии левой кнопки, но не для последующих движений. Аналогичным образом **MOUSEEVENTF_LEFTUP** устанавливается только при первом освобождении кнопки.

Нельзя одновременно указать флаг **MOUSEEVENTF_WHEEL** и флаг **MOUSEEVENTF_XDOWN** или **MOUSEEVENTF_XUP** в параметре **dwFlags** , так как для них обоих требуется использовать поле **mouseData** .

Значение	Значение
MOUSEEVENTF_MOVE 0x0001	Произошло перемещение.
MOUSEEVENTF_LEFTDOWN 0x0002	Нажата левая кнопка.
MOUSEEVENTF_LEFTUP 0x0004	Левая кнопка была отпущена.
MOUSEEVENTF_RIGHTDOWN 0x0008	Нажата правая кнопка.
MOUSEEVENTF_RIGHTUP 0x0010	Отпущена правая кнопка.
MOUSEEVENTF_MIDDLEDOWN 0x0020	Нажата средняя кнопка.
MOUSEEVENTF_MIDDLEUP 0x0040	Средняя кнопка была отпущена.
MOUSEEVENTF_XDOWN 0x0080	Нажата кнопка X.
MOUSEEVENTF_XUP 0x0100	Кнопка X была отпущена.
MOUSEEVENTF_WHEEL 0x0800	Колесо было перемещено, если у мыши есть колесо. Величина перемещения указывается в <code>mouseData</code> .
MOUSEEVENTF_HWHEEL 0x1000	Колесо было перемещено по горизонтали, если у мыши есть колесо. Величина перемещения указывается в <code>mouseData</code> . <small>Windows XP/2000:</small> это значение не поддерживается.
MOUSEEVENTF_MOVE_NOCOALESCE 0x2000	Сообщения WM_MOUSEMOVE не будут объединяться. Поведение по умолчанию — объединение <code>WM_MOUSEMOVE</code> сообщений. <small>Windows XP/2000:</small> это значение не поддерживается.
MOUSEEVENTF_VIRTUALDESK 0x4000	Сопоставляет координаты со всем рабочим столом. Должен использоваться с MOUSEEVENTF_ABSOLUTE .
MOUSEEVENTF_ABSOLUTE 0x8000	Элементы <code>dx</code> и <code>dy</code> содержат нормализованные абсолютные координаты. Если флаг не задан, <code>dx</code> и <code>dy</code> содержат относительные данные (изменение позиции с момента последнего сообщаемого положения). Этот флаг можно установить или не задать, независимо от типа мыши или другого указывающего устройства, если таковое имеется, подключенного к системе.

Значение	Значение
	Дополнительные сведения об относительном движении мыши см. в следующем разделе Примечания.

time

Тип: **DWORD**

Метка времени для события в миллисекундах. Если этот параметр равен 0, система предоставит собственную метку времени.

dwExtraInfo

Тип: **ULONG_PTR**

Дополнительное значение, связанное с событием мыши. Приложение вызывает [GetMessageExtraInfo](#) для получения этих дополнительных сведений.

Комментарии

Если мышь переместилась, **MOUSEEVENTF_MOVE**, dx и dy указывают сведения об этом **перемещении**. Сведения указываются как абсолютные или относительные целочисленные значения.

Если указано **MOUSEEVENTF_ABSOLUTE** значение, dx и dy содержат нормализованные абсолютные координаты от 0 до 65 535. Процедура события сопоставляет эти координаты с поверхностью дисплея. Координаты (0,0) сопоставляется с левым верхним углом поверхности дисплея; координаты (65535,65535) сопоставляется с правым нижним углом. В мультимониторной системе координаты сопоставляются с основным монитором.

Если указано **MOUSEEVENTF_VIRTUALDESK**, координаты сопоставляются со всем виртуальным рабочим столом.

Если значение **MOUSEEVENTF_ABSOLUTE** не указано, dx и dy указывают перемещение относительно предыдущего события мыши (последней сообщаемой позиции). Положительные значения означают перемещение мыши вправо (или вниз); Отрицательные значения означают, что мышь переместилась влево (или вверх).

Относительное движение мыши зависит от влияния скорости мыши и пороговых значений двух мышей. Пользователь задает эти три значения с помощью ползунка

Скорость указателя на странице свойств мыши панель управления. Эти значения можно получить и задать с помощью функции [SystemParametersInfo](#).

Система применяет два теста к указанному относительному перемещению мыши. Если указанное расстояние по оси x или y больше первого порогового значения мыши, а скорость мыши не равна нулю, система удваивает расстояние. Если указанное расстояние вдоль оси x или y больше второго порогового значения мыши, а скорость мыши равна двум, система удваивает расстояние, полученное в результате применения первого порогового теста. Таким образом, система может умножить указанное относительное перемещение мыши по оси x или y до четырех раз.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

[Основные понятия](#)

[GetMessageExtraInfo](#)

[INPUT](#)

[Ввод с клавиатуры](#)

[Справочные материалы](#)

[SendInput](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура MOUSEMOVEPOINT (winuser.h)

Статья 28.08.2023

Содержит сведения о расположении мыши в координатах экрана.

Синтаксис

C++

```
typedef struct tagMOUSEMOVEPOINT {
    int          x;
    int          y;
    DWORD        time;
    ULONG_PTR    dwExtraInfo;
} MOUSEMOVEPOINT, *PMOUSEMOVEPOINT, *LPMOUSEMOVEPOINT;
```

Члены

x

Тип: **int**

Координата мыши по оси X.

y

Тип: **int**

Координата мыши по оси Y.

time

Тип: **DWORD**

Метка времени координаты мыши.

dwExtraInfo

Тип: **ULONG_PTR**

Дополнительные сведения, связанные с этой координатой.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

[Основные понятия](#)

[GetMouseMovePointsEx](#)

[Ввод с помощью мыши](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Макрос NEXTRAWINPUTBLOCK (winuser.h)

Статья 15.03.2023

Извлекает расположение следующей структуры в массиве структур [RAWINPUT](#).

Синтаксис

C++

```
void NEXTRAWINPUTBLOCK(  
    ptr  
) ;
```

Параметры

ptr

Указатель на структуру в массиве структур [RAWINPUT](#).

Возвращаемое значение

None

Remarks

Этот макрос вызывается многократно для обхода массива структур [RAWINPUT](#).

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)

См. также раздел

[Основные понятия](#)

[RAWINPUT](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция OemKeyScan (winuser.h)

Статья 27.08.2023

Сопоставляет коды OEMASCII от 0 до 0x0FF с кодами сканирования OEM и состояниями сдвига. Функция предоставляет сведения, позволяющие программе отправлять текст OEM в другую программу путем имитации ввода с клавиатуры.

Синтаксис

C++

```
DWORD OemKeyScan(  
    [in] WORD wOemChar  
);
```

Параметры

[in] wOemChar

Тип: WORD

Значение ASCII для символа OEM.

Возвращаемое значение

Тип: DWORD

Слово низкого порядка возвращаемого значения содержит код сканирования символа OEM, а слово высокого порядка — состояние сдвига, которое может быть сочетанием следующих битов.

bit	Описание
1	Нажата любая клавиша SHIFT.
2	Нажата либо клавиша CTRL.
4	Нажата клавиша ALT.
8	Нажата клавиша Ханкаку.
16	Зарезервировано (определяется драйвером раскладки)

клавиатуры).

32

Зарезервировано (определяется драйвером раскладки клавиатуры).

Если символ не может быть создан одним нажатием клавиши с помощью текущей раскладки клавиатуры, возвращается значение -1.

Комментарии

Эта функция не предоставляет переводы для символов, для которых требуются клавиши CTRL+ALT или неработающие клавиши. Символы, не переведенные этой функцией, должны копироваться путем имитации ввода с помощью механизма клавиатуры ALT+. Ключ NUMLOCK должен быть отключен.

Эта функция не предоставляет переводы для символов, которые нельзя ввести одним нажатием клавиши с помощью текущей раскладки клавиатуры, например символы с диакритических знаков, требующих неработающих клавиш. Символы, не переведенные этой функцией, можно смоделировать с помощью механизма клавиатуры ALT+. Ключ NUMLOCK должен быть включен.

Эта функция реализуется с помощью функции [VkKeyScan](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с клавиатуры](#)

[Справочные материалы](#)

[VkKeyScan](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура RAWHID (winuser.h)

Статья 28.08.2023

Описывает формат необработанных входных данных с устройства HID.

Синтаксис

C++

```
typedef struct tagRAWHID {
    DWORD dwSizeHid;
    DWORD dwCount;
    BYTE bRawData[1];
} RAWHID, *PRAWHID, *LPRAWHID;
```

Члены

`dwSizeHid`

Тип: **DWORD**

Размер (в байтах) всех входных данных HID в `bRawData`.

`dwCount`

Тип: **DWORD**

Количество входных данных HID в `bRawData`.

`bRawData[1]`

Тип: **BYTE[1]**

Необработанные входные данные в виде массива байтов.

Комментарии

Каждая [WM_INPUT](#) может указывать на несколько входных данных, но все входные данные поступают из одного HID. Размер массива `bRawData` — `dwSizeHid * dwCount`.

Дополнительные сведения см. в разделе [Интерпретация отчетов HID](#).

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[RAWINPUT](#)

[Необработанные входные данные](#)

[Общие сведения об устройствах hid](#)

Справочные материалы

[WM_INPUT](#)

[Интерпретация отчетов HID](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура RAWINPUT (winuser.h)

Статья 01.06.2023

Содержит необработанные входные данные с устройства.

Синтаксис

C++

```
typedef struct tagRAWINPUT {
    RAWINPUTHEADER header;
    union {
        RAWMOUSE     mouse;
        RAWKEYBOARD keyboard;
        RAWHID       hid;
    } data;
} RAWINPUT, *PRAWINPUT, *LPRAWINPUT;
```

Члены

header

Тип: [RAWINPUTHEADER](#)

Необработанные входные данные.

data

data.mouse

Тип: [RAWMOUSE](#)

Если данные поступают с мыши, это необработанные входные данные.

data.keyboard

Тип: [RAWKEYBOARD](#)

Если данные поступают с клавиатуры, это необработанные входные данные.

data.hid

Тип: [RAWHID](#)

Если данные поступают из HID, это необработанные входные данные.

Комментарии

Дескриптор этой структуры передается в параметре *lParamWM_INPUT*.

Чтобы получить подробные сведения, такие как заголовок и содержимое необработанных входных данных, вызовите [Метод GetRawInputData](#).

Чтобы считать RAWINPUT в цикле сообщений в качестве буферизованного чтения, вызовите [Метод GetRawInputBuffer](#).

Чтобы получить сведения об устройстве, вызовите [Метод GetRawInputDeviceInfo](#) с *hDevice* из [RAWINPUTHEADER](#).

Необработанные входные данные доступны, только если приложение вызывает [RegisterRawInputDevices](#) с допустимыми спецификациями устройства.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

[Основные понятия](#)

[GetRawInputBuffer](#)

[GetRawInputData](#)

[RAWHID](#)

[RAWINPUTHEADER](#)

[RAWKEYBOARD](#)

[RAWMOUSE](#)

[Необработанные входные данные](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура RAWINPUTDEVICE (winuser.h)

Статья 28.08.2023

Определяет сведения для необработанных устройств ввода.

Синтаксис

C++

```
typedef struct tagRAWINPUTDEVICE {  
    USHORT usUsagePage;  
    USHORT usUsage;  
    DWORD dwFlags;  
    HWND hwndTarget;  
} RAWINPUTDEVICE, *PRAWINPUTDEVICE, *LPRAWINPUTDEVICE;
```

Члены

usUsagePage

Тип: **USHORT**

Страница "Использование" коллекции верхнего уровня для необработанного устройства ввода. Дополнительные сведения о возможных значениях см. [в разделе HID-клиенты, поддерживаемые в Windows](#).

usUsage

Тип: **USHORT**

ИД использования коллекции верхнего уровня для необработанного устройства ввода. Дополнительные сведения о возможных значениях см. [в разделе HID-клиенты, поддерживаемые в Windows](#).

dwFlags

Тип: **DWORD**

Флаг режима, указывающий способ интерпретации сведений, предоставляемых **usUsagePage** и **usUsage**. Это может быть ноль (по умолчанию) или одно из следующих значений. По умолчанию операционная система отправляет

необработанные входные данные с устройств с указанной [коллекцией верхнего уровня](#) (TLC) в зарегистрированное приложение при условии, что оно имеет фокус окна.

Значение	Значение
RIDEV_REMOVE 0x00000001	Если этот параметр задан, коллекция верхнего уровня удаляется из списка включения. Это указывает операционной системе прекратить чтение с устройства, соответствующего коллекции верхнего уровня.
RIDEV_EXCLUDE 0x00000010	Если задано значение , это указывает коллекции верхнего уровня, которые необходимо исключить при чтении полной страницы использования. Этот флаг влияет только на TLC, страница использования которого уже указана с помощью RIDEV_PAGEONLY.
RIDEV_PAGEONLY 0x00000020	Если задано значение , это означает, что все устройства, коллекция верхнего уровня которых является коллекцией из указанного объекта usUsagePage . Обратите внимание, что usUsage должно быть равно нулю. Чтобы исключить определенную коллекцию верхнего уровня, используйте RIDEV_EXCLUDE.
RIDEV_NOLEGACY 0x00000030	Если этот параметр задан, это не позволит любым устройствам, указанным usUsagePage или usUsage , создавать устаревшие сообщения . Это касается только мыши и клавиатуры. См. заметки.
RIDEV_INPUTSINK 0x00000100	Если этот параметр задан, вызывающий объект может получать входные данные, даже если вызывающий объект не находится на переднем плане. Обратите внимание, что необходимо указать hwndTarget .
RIDEV_CAPTUREMOUSE 0x00000200	Если этот параметр задан, нажатие кнопки мыши не активирует другое окно. RIDEV_CAPTUREMOUSE можно указать только в том случае, если для устройства с мышью задано RIDEV_NOLEGACY .
RIDEV_NOHOTKEYS 0x00000200	Если этот параметр задан, клавиши клавиатуры, определенные приложением, не обрабатываются. Тем не менее, системные горячие клавиши; например, ALT+TAB и CTRL+ALT+DEL по-прежнему обрабатываются. По умолчанию обрабатываются все клавиши клавиатуры. RIDEV_NOHOTKEYS можно указать, даже если RIDEV_NOLEGACY не указан, а hwndTarget имеет значение NULL.

RIDEV_APPKEYS 0x00000400	Если этот параметр задан, то обрабатываются клавиши команд приложения. RIDEV_APPKEYS можно указать только в том случае, если для устройства клавиатуры задано RIDEV_NOLEGACY .
RIDEV_EXINPUTSINK 0x00001000	Если задано значение , это позволяет вызывающему объекту получать входные данные в фоновом режиме только в том случае, если приложение переднего плана не обрабатывает их. Иными словами, если приложение переднего плана не зарегистрировано для необработанных входных данных, то зарегистрированное фоновое приложение получит входные данные. Windows XP: Этот флаг не поддерживается до Windows Vista
RIDEV_DEVNOTIFY 0x00002000	Если этот параметр задан, вызывающий абонент сможет получать WM_INPUT_DEVICE_CHANGE уведомления о поступлении и удалении устройства. Windows XP: Этот флаг не поддерживается до Windows Vista

hwndTarget

Тип: **HWND**

Дескриптор целевого окна. Если значение **РАВНО NULL**, оно следует за фокусом клавиатуры.

Комментарии

Если **RIDEV_NOLEGACY** задано для мыши или клавиатуры, система не создает устаревшее сообщение для этого устройства для приложения. Например, если для параметра TLC мыши задано **RIDEV_NOLEGACY**, **WM_LBUTTONDOWN** и [связанные с ними устаревшие сообщения мыши](#) не создаются. Аналогичным образом, если для клавиатуры для TLC задано **RIDEV_NOLEGACY**, **WM_KEYDOWN** и [связанные с ними устаревшие сообщения клавиатуры](#) не создаются.

Если задано **RIDEV_REMOVE** и член **hwndTarget** не имеет значения **NULL**, функция [RegisterRawInputDevices](#) завершится ошибкой.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[GetRegisteredRawInputDevices](#)

[Необработанные входные данные](#)

[Общие сведения об устройствах с человеческим интерфейсом \(HID\)](#)

[Клиенты HID, поддерживаемые в Windows](#)

[Домашняя страница HID USB ↗](#)

Справочные материалы

[RegisterRawInputDevices](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура RAWINPUTDEVICELIST (winuser.h)

Статья 28.08.2023

Содержит сведения о необработанном устройстве ввода.

Синтаксис

C++

```
typedef struct tagRAWINPUTDEVICELIST {  
    HANDLE hDevice;  
    DWORD dwType;  
} RAWINPUTDEVICELIST, *PRAWINPUTDEVICELIST;
```

Члены

`hDevice`

Тип: **HANDLE**

Дескриптор для необработанного устройства ввода.

`dwType`

Тип: **DWORD**

Тип устройства. Это может быть одно из следующих значений.

Значение	Значение
RIM_TYPEHID 2	Устройство — это hid, который не является клавиатурой и не мышью.
RIM_TYPEKEYBOARD 1	Устройство является клавиатурой.
RIM_TYPEMOUSE 0	Устройством является мышь.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[GetRawInputDeviceList](#)

[Необработанные входные данные](#)

Справочные материалы

Обратная связь

Были ли сведения на этой странице полезными?

 Да	 Нет
--	--

[Получить справку в Microsoft Q&A](#)

Структура RAWINPUTHEADER (winuser.h)

Статья 28.08.2023

Содержит сведения о заголовке, которые являются частью необработанных входных данных.

Синтаксис

C++

```
typedef struct tagRAWINPUTHEADER {
    DWORD dwType;
    DWORD dwSize;
    HANDLE hDevice;
    WPARAM wParam;
} RAWINPUTHEADER, *PRAWINPUTHEADER, *LPRAWINPUTHEADER;
```

Члены

`dwType`

Тип: **DWORD**

Тип необработанных входных данных. Может иметь одно из следующих значений.

Значение	Значение
RIM_TYPEMOUSE 0	Необработанные входные данные поступают от мыши.
RIM_TYPEKEYBOARD 1	Необработанный ввод поступает с клавиатуры.
RIM_TYPEHID 2	Необработанный ввод поступает с некоторых устройств, которые не являются клавиатурой или мышью.

`dwSize`

Тип: **DWORD**

Размер всего входного пакета данных (в байтах). К ним относятся [RAWINPUT](#) и возможные дополнительные входные отчеты в массиве переменной длины [RAWHID](#).

`hDevice`

Тип: `HANDLE`

Дескриптор устройства, создающего необработанные входные данные.

`wParam`

Тип: `WPARAM`

Значение, переданное в параметре `wParam` сообщения [WM_INPUT](#).

Комментарии

Чтобы получить дополнительные сведения об устройстве, используйте `hDevice` в вызове [GetRawDeviceInfo](#). Значение `hDevice` может быть равно нулю, если входные данные получены с сенсорной панели точности.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[GetRawDeviceInfo](#)

[Структура RAWINPUT](#)

[Структура RAWKEYBOARD](#)

[Структура RAWMOUSE](#)

[Структура RAWHID](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура RAWKEYBOARD (winuser.h)

Статья 28.08.2023

Содержит сведения о состоянии клавиатуры.

Синтаксис

C++

```
typedef struct tagRAWKEYBOARD {
    USHORT MakeCode;
    USHORT Flags;
    USHORT Reserved;
    USHORT VKey;
    UINT   Message;
    ULONG  ExtraInformation;
} RAWKEYBOARD, *PRAWKEYBOARD, *LPRAWKEYBOARD;
```

Члены

MakeCode

Тип: **USHORT**

Указывает код сканирования, связанный с нажатием клавиши. См. заметки.

Flags

Тип: **USHORT**

Флаги для сведений о коде сканирования. Это может быть один или несколько из следующих вариантов:

Значение	Значение
RI_KEY_MAKE 0	Ключ не работает.
RI_KEY_BREAK 1	Ключ включен.
RI_KEY_E0 2	Код сканирования имеет префикс E0.
RI_KEY_E1 4	Код сканирования имеет префикс E1.

Reserved

Тип: **USHORT**

Защищены; значение должно быть равно нулю.

VKey

Тип: **USHORT**

Соответствующий устаревший код виртуального ключа.

Message

Тип: **UINT**

Соответствующее сообщение окна клавиатуры прежних версий, например [WM_KEYDOWN](#), [WM_SYSKEYDOWN](#) и т. д.

ExtraInformation

Тип: **ULONG**

Дополнительные сведения о событии, относящиеся к конкретному устройству.

Комментарии

Список значений **MakeCode** приведен в [обзоре ввода с помощью клавиатуры](#) (см . раздел Проверка 1 в столбце make).

Для клавиатуры HID значения **MakeCode** создаются [драйвером сопоставителя клиента HID](#) , который преобразует использование HID в коды сканирования в соответствии с [USB HID в таблицу перевода кода сканирования PS/2](#) (см. ps/2 Set 1 Make column).

KEYBOARD_OVERRUN_MAKE_CODE — это специальное значение **MakeCode** , которое отправляется, если нажата недопустимая или неузнаваемая комбинация клавиш или количество нажатых клавиш превышает ограничение для этой клавиатуры.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]

См. также раздел

- [GetRawInputDeviceInfo](#)
- [RAWINPUT](#)
- [Необработанные входные данные](#)
- [Клиентские драйверы HID клавиатуры и мыши](#)
- [Таблица преобразования кода сканирования с USB HID на PS/2 ↗](#)
- [структура KEYBOARD_INPUT_DATA](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура RAWMOUSE (winuser.h)

Статья 28.08.2023

Содержит сведения о состоянии мыши.

Синтаксис

C++

```
typedef struct tagRAWMOUSE {
    USHORT usFlags;
    union {
        ULONG ulButtons;
        struct {
            USHORT usButtonFlags;
            USHORT usButtonData;
        } DUMMYSTRUCTNAME;
    } DUMMYUNIONNAME;
    ULONG ulRawButtons;
    LONG lLastX;
    LONG lLastY;
    ULONG ulExtraInformation;
} RAWMOUSE, *PRAWMOUSE, *LPRAWMOUSE;
```

Члены

usFlags

Тип: **USHORT**

Состояние мыши. Этот элемент может быть любым разумным сочетанием следующего.

Значение	Значение
MOUSE_MOVE_RELATIVE 0x00	Данные о перемещении мыши относительно последней позиции мыши. Дополнительные сведения о перемещении мыши см. в следующем разделе Примечаний.
MOUSE_MOVE_ABSOLUTE 0x01	Данные о перемещении мыши основаны на абсолютном положении. Дополнительные сведения о перемещении мыши см. в следующем разделе Примечаний.

Значение	Значение
MOUSE_VIRTUAL_DESKTOP0x02	Координаты мыши сопоставляются с виртуальным рабочим столом (для системы с несколькими мониторами). Дополнительные сведения о перемещении мыши см. в следующем разделе Примечаний.
MOUSE_ATTRIBUTES_CHANGED0x04	Изменены атрибуты мыши; приложению необходимо запрашивать атрибуты мыши.
MOUSE_MOVE_NOCOALESCE0x08	Это событие перемещения мыши не было объединено. События перемещения мыши можно объединять по умолчанию. Windows XP/2000: это значение не поддерживается.

DUMMYUNIONNAME

DUMMYUNIONNAME.ulButtons

Тип: **ULONG**

Зарезервировано.

DUMMYUNIONNAME.DUMMYSTRUCTNAME

DUMMYUNIONNAME.DUMMYSTRUCTNAME.usButtonFlags

Тип: **USHORT**

Состояние перехода кнопок мыши. Этот элемент может быть одним или несколькими из следующих значений.

Значение	Значение
RI_MOUSE_BUTTON_1_DOWN RI_MOUSE_LEFT_BUTTON_DOWN0x0001	Левая кнопка была изменена на вниз.
RI_MOUSE_BUTTON_1_UP RI_MOUSE_LEFT_BUTTON_UP0x0002	Левая кнопка была изменена на "Вверх".
RI_MOUSE_BUTTON_2_DOWN RI_MOUSE_RIGHT_BUTTON_DOWN0x0004	Правая кнопка была изменена на вниз.
RI_MOUSE_BUTTON_2_UP	Правая кнопка изменена на "Вверх".

Значение	Значение
RI_MOUSE_RIGHT_BUTTON_UP0x0008	
RI_MOUSE_BUTTON_3_DOWN RI_MOUSE_MIDDLE_BUTTON_DOWN0x0010	Средняя кнопка изменена на вниз.
RI_MOUSE_BUTTON_3_UP RI_MOUSE_MIDDLE_BUTTON_UP0x0020	Средняя кнопка изменена на вверх.
RI_MOUSE_BUTTON_4_DOWN0x0040	XBUTTON1 изменено на down.
RI_MOUSE_BUTTON_4_UP0x0080	XBUTTON1 изменено на .
RI_MOUSE_BUTTON_5_DOWN0x0100	XBUTTON2 изменена на неупустимая.
RI_MOUSE_BUTTON_5_UP0x0200	XBUTTON2 изменено на .
RI_MOUSE_WHEEL0x0400	Необработанные входные данные поступают от колесика мыши. Разностное колесо хранится в usButtonData. Положительное значение указывает, что колесико было повернуто вперед, от пользователя; отрицательное значение указывает, что колесико было повернуло назад к пользователю. Дополнительные сведения см. в следующем разделе Примечаний.
RI_MOUSE_HWHEEL0x0800	Необработанные входные данные поступают от горизонтального колесика мыши. Разностное колесо хранится в usButtonData. Положительное значение указывает, что колесо повернулось вправо; отрицательное значение указывает, что колесико повернулось влево. Дополнительные сведения см. в следующем разделе Примечаний. Windows XP/2000: это значение не поддерживается.

DUMMYUNIONNAME.DUMMYSTRUCTNAME.usButtonData

Тип: USHORT

Если параметр usButtonFlags имеет RI_MOUSE_WHEEL или RI_MOUSE_HWHEEL, этот элемент указывает расстояние, на которое поворачивается колесико. Дополнительные сведения см. в следующем разделе Примечаний.

ulRawButtons

Тип: **ULONG**

Необработанное состояние кнопок мыши. Подсистема Win32 не использует этот элемент.

lLastX

Тип: **LONG**

Движение в направлении X. Это знаковое относительное движение или абсолютное движение в зависимости от значения usFlags.

lLastY

Тип: **LONG**

Движение в направлении Y. Это знаковое относительное движение или абсолютное движение в зависимости от значения usFlags.

ulExtraInformation

Тип: **ULONG**

Дополнительные сведения об устройстве для события.

Комментарии

Если мышь перемещена, обозначенная MOUSE_MOVE_RELATIVE или MOUSE_MOVE_ABSOLUTE, lLastX и lLastY указывают сведения об этом перемещении. Сведения указываются как относительные или абсолютные целочисленные значения.

Если указано MOUSE_MOVE_RELATIVE значение, lLastX и lLastY указывают перемещение относительно предыдущего события мыши (последней указанной позиции). Положительные значения означают перемещение мыши вправо (или вниз); Отрицательные значения означают перемещение мыши влево (или вверх).

Если указано MOUSE_MOVE_ABSOLUTE значение, lLastX и lLastY содержат нормализованные абсолютные координаты в диапазоне от 0 до 65 535. Координата

(0,0) сопоставляется с левым верхним углом поверхности дисплея; координаты (65535,65535) сопоставляется с правым нижним углом. В мультимониторной системе координаты сопоставляются с основным монитором.

Если **MOUSE_VIRTUAL_DESKTOP** указан в дополнение к **MOUSE_MOVE_ABSOLUTE**, координаты сопоставляются со всем виртуальным рабочим столом.

C++

```
if ((rawMouse.usFlags & MOUSE_MOVE_ABSOLUTE) == MOUSE_MOVE_ABSOLUTE)
{
    bool isVirtualDesktop = (rawMouse.usFlags & MOUSE_VIRTUAL_DESKTOP) ==
MOUSE_VIRTUAL_DESKTOP;

    int width = GetSystemMetrics(isVirtualDesktop ? SM_CXVIRTUALSCREEN :
SM_CXSCREEN);
    int height = GetSystemMetrics(isVirtualDesktop ? SM_CYVIRTUALSCREEN :
SM_CYSCREEN);

    int absoluteX = int((rawMouse.lLastX / 65535.0f) * width);
    int absoluteY = int((rawMouse.lLastY / 65535.0f) * height);
}
else if (rawMouse.lLastX != 0 || rawMouse.lLastY != 0)
{
    int relativeX = rawMouse.lLastX;
    int relativeY = rawMouse.lLastY;
}
```

В отличие от устаревших **WM_MOUSEMOVE** оконных сообщений о событиях необработанных входных данных мыши не зависит от скорости мыши, заданной на странице **свойств мыши** панель управления. Дополнительные сведения см. в разделе [Общие сведения о вводе с помощью мыши](#).

Если колесико мыши перемещено, указанное **RI_MOUSE_WHEEL** или **RI_MOUSE_HWHEEL** в **usButtonFlags**, **usButtonData** содержит **короткое** значение со знаком, указывающее расстояние поворота колесика.

Поворот колесика будет кратным **WHEEL_DELTA**, который имеет значение 120. Это пороговое значение для выполняемого действия, и одно такое действие (например, прокрутка на один шаг) должно выполняться для каждой дельты.

Для разностного значения задано значение 120, чтобы корпорация Майкрософт или другие поставщики могли создавать колеса с более тонким разрешением (свободно вращающееся колесо без вырезов), чтобы отправлять больше сообщений на поворот, но с меньшим значением в каждом сообщении. Чтобы использовать эту функцию, можно добавить входящие разностные значения до **достижения WHEEL_DELTA** (чтобы при разностном повороте вы получили тот же

ответ), или прокручивать частичные строки в ответ на более частые сообщения. Вы также можете выбрать степень детализации прокрутки и накапливать разностные значения, пока она не будет достигнута.

Приложение также может получить текущие пользовательские параметры "строки для прокрутки" и "символы для прокрутки", используя API [SystemParametersInfo](#) с параметром **SPI_GETWHEELSCROLLLINES** или **SPI_GETWHEELSCROLLCHARS**.

Ниже приведен пример такого кода обработки колес:

C++

```
if ((rawMouse.usButtonFlags & RI_MOUSE_WHEEL) == RI_MOUSE_WHEEL ||  
    (rawMouse.usButtonFlags & RI_MOUSE_HWHEEL) == RI_MOUSE_HWHEEL)  
{  
    static const unsigned long defaultScrollLinesPerWheelDelta = 3;  
    static const unsigned long defaultScrollCharsPerWheelDelta = 1;  
  
    float wheelDelta = (float)(short)rawMouse.usButtonData;  
    float numTicks = wheelDelta / WHEEL_DELTA;  
  
    bool isHorizontalScroll = (rawMouse.usButtonFlags & RI_MOUSE_HWHEEL) ==  
RI_MOUSE_HWHEEL;  
    bool isScrollByPage = false;  
    float scrollDelta = numTicks;  
  
    if (isHorizontalScroll)  
    {  
        unsigned long scrollChars = defaultScrollCharsPerWheelDelta;  
        SystemParametersInfo(SPI_GETWHEELSCROLLCHARS, 0, &scrollChars, 0);  
        scrollDelta *= scrollChars;  
    }  
    else  
    {  
        unsigned long scrollLines = defaultScrollLinesPerWheelDelta;  
        SystemParametersInfo(SPI_GETWHEELSCROLLLINES, 0, &scrollLines, 0);  
        if (scrollLines == WHEEL_PAGESCROLL)  
            isScrollByPage = true;  
        else  
            scrollDelta *= scrollLines;  
    }  
}
```

Требования

Минимальная версия клиента

Windows XP [только классические приложения]

Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[GetRawInputDeviceInfo](#)

[RAWINPUT](#)

[Необработанные входные данные](#)

Справочные материалы

[Структура MOUSEINPUT](#)

[Функция SendInput](#)

[структура MOUSE_INPUT_DATA](#)

[Общие сведения о вводе с помощью мыши \(прежние версии\)](#)

[Уведомления о вводе с помощью мыши \(прежние версии\)](#)

[SystemParametersInfo](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция RegisterForTooltipDismissNotification (winuser.h)

Статья 15.08.2023

Регистрирует или отменяет регистрацию окон для получения уведомлений о закрытии окон всплывающих подсказок.

Синтаксис

C++

```
BOOL RegisterForTooltipDismissNotification(
    HWND           hWnd,
    TOOLTIP_DISMISS_FLAGS tdFlags
);
```

Параметры

hWnd

Тип: **HWND**

Дескриптор окна для получения сообщения **WM_TOOLTIPDISMISS**.

tdFlags

Тип: **TOOLTIP_DISMISS_FLAGS**

Значение перечисления , указывающее, регистрирует ли функция или отменяет регистрацию окна. **TDF_REGISTER** для регистрации; **TDF_UNREGISTER** отменить регистрацию.

Возвращаемое значение

ЗНАЧЕНИЕ TRUE , если окно было успешно зарегистрировано или отменено; в противном случае — **FALSE**. (См. раздел «Примечания».)

Комментарии

Эта функция делает подсказки более доступными, позволяя приложениям и платформам, поддерживающим всплывающие подсказки, регистрировать и отменять регистрацию с помощью **сообщения WM_TOOLTIPDISMISS**, когда системе требуется закрыть все всплывающие подсказки.

Приложения должны регистрироваться для получения этого уведомления каждый раз, когда они отображают подсказку и скрывают свои подсказки в ответ на **WM_TOOLTIPDISMISS** сообщение. Если подсказка скрыта по какой-либо другой причине, например для действия мыши, приложение должно отменить регистрацию.

Системные триггеры для закрытия подсказки включают одиночную клавишу CTRL вверх или CTRL+SHIFT+F10. (Набор триггеров может меняться со временем.)

Функция принимает **HWND** окна подсказки или **HWND** окна приложения с дочерними подсказками.

- Если сама подсказка **HWND** зарегистрирована, окно подсказки должно зарегистрироваться при отображении и закрыть при получении **сообщения WM_TOOLTIPDISMISS**.
- Если приложение **HWND** регистрируется от имени своих подсказок, окно приложения должно зарегистрироваться при отображении всплывающих подсказок и закрыть все подсказки при получении **сообщения WM_TOOLTIPDISMISS**.

Ожидается, что подсказки или окна приложений будут вызывать функцию для регистрации при каждом отображении подсказок. Зарегистрированные окна автоматически отменяются при публикации **WM_TOOLTIPDISMISS**.

Флаг **TDF_UNREGISTER** используется для явной отмены регистрации окна, когда окно подсказки закрывается прерогативой приложения или платформы (например, перемещение курсора из "безопасной зоны"). Если приложение или платформа вызывает `RegisterForTooltipDismissNotification` с **TDF_UNREGISTER** после автоматической отмены регистрации окна, функция возвращает значение **FALSE**. Это не влияет на будущие регистрации.

Возвращаемые значения

HWND, переданный в функцию, должен принадлежать вызывающей процедуре; В противном случае функция возвращает значение **FALSE**.

При вызове с **TDF_REGISTER** и окном, принадлежащим вызывающей процедуре, функция возвращает значение **TRUE**, если окно было успешно зарегистрировано, или **FALSE**, если окно уже зарегистрировано. Окно обрабатывается как зарегистрированное в любом случае.

При вызове с **TDF_UNREGISTER** и окнами, принадлежащими вызывающей процедуре, функция возвращает значение **TRUE**, если окно успешно отменено, или значение **FALSE**, если окна не были зарегистрированы в данный момент. Окно обрабатывается как незарегистрированное в любом случае.

Требования

Минимальная версия клиента	сборка Windows 11 22621
Целевая платформа	Windows
Header	winuser.h
Библиотека	user32.lib

См. также раздел

[Подсказка](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция RegisterHotKey (winuser.h)

Статья 27.08.2023

Определяет горячий ключ на уровне системы.

Синтаксис

C++

```
BOOL RegisterHotKey(
    [in, optional] HWND hWnd,
    [in]           int id,
    [in]           UINT fsModifiers,
    [in]           UINT vk
);
```

Параметры

[in, optional] hWnd

Тип: HWND

Дескриптор окна, который будет получать WM_HOTKEY сообщения, созданные горячей клавишей. Если этот параметр имеет значение NULL, WM_HOTKEY сообщения отправляются в очередь сообщений вызывающего потока и должны обрабатываться в цикле сообщений.

[in] id

Тип: int

Идентификатор горячего ключа. Если параметр hWnd имеет значение NULL, то горячий ключ связывается с текущим потоком, а не с определенным окном. Если горячий ключ уже существует с теми же параметрами hWnd и id , см. примечания о выполненных действиях.

[in] fsModifiers

Тип: UINT

Клавиши, которые должны быть нажаты в сочетании с клавишей, указанной параметром vk , чтобы создать сообщение WM_HOTKEY . Параметр fsModifiers может быть сочетанием следующих значений.

Значение	Значение
MOD_ALT 0x0001	Обе клавиши ALT должны быть удержаны.
MOD_CONTROL 0x0002	Прижата любая клавиша CTRL.
MOD_NO_REPEAT 0x4000	Изменяет поведение клавиши так, чтобы автоматический повтор клавиатуры не возвращал несколько уведомлений с помощью клавиш горячей клавиши. Windows Vista: Этот флаг не поддерживается.
MOD_SHIFT 0x0004	Обе клавиши SHIFT должны быть удерживаются.
MOD_WIN 0x0008	Любая клавиша WINDOWS была удержана. Эти клавиши помечены логотипом Windows. Сочетания клавиш, включающие клавишу WINDOWS, зарезервированы для использования операционной системой.

[in] vk

Тип: **UINT**

Код виртуального ключа горячего ключа. См . [раздел Коды виртуальных ключей](#).

Возвращаемое значение

Тип: **BOOL**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

При нажатии клавиши система ищет совпадение со всеми горячими клавишами. После поиска совпадения система отправляет [сообщение WM_HOTKEY](#) в очередь сообщений окна, с которым связана горячая клавиша. Если горячая клавиша не связана с окном, [сообщение WM_HOTKEY](#) будет отправлено в поток, связанный с горячей клавишей.

Эта функция не может связать горячий ключ с окном, созданным другим потоком.

RegisterHotKey завершается ошибкой , если нажатия клавиш, указанные для горячего ключа, уже зарегистрированы другим горячим ключом.

Если горячий ключ уже существует с теми же параметрами *hWnd* и *id* , он сохраняется вместе с новым горячим ключом. Приложение должно явно вызвать [UnregisterHotKey](#) , чтобы отменить регистрацию старого горячего ключа.

Windows Server 2003: Если горячий ключ уже существует с теми же параметрами *hWnd* и *id* , он заменяется новым горячим ключом.

Ключ F12 зарезервирован для постоянного использования отладчиком, поэтому его не следует регистрировать как горячий ключ. Даже если вы не выполняете отладку приложения, F12 резервируется на случай, если отладчик режима ядра или JIT-отладчик является резидентом.

Приложение должно указать значение идентификатора в диапазоне 0x0000 по 0xBFFF. Общая библиотека DLL должна указывать значение в диапазоне 0xC000 по 0xFFFF (диапазон, возвращаемый функцией [GlobalAddAtom](#)). Чтобы избежать конфликтов с идентификаторами горячих ключей, определенными другими общими библиотеками DLL, библиотека DLL должна использовать функцию [GlobalAddAtom](#) для получения идентификатора горячего ключа.

Примеры

В следующем примере показано, как использовать функцию **RegisterHotKey** с флагом **MOD_NOREPEAT** . В этом примере для потока main регистрируется горячая клавиша ALT+b. После нажатия клавиши поток получит **WM_HOTKEY** сообщение, которое будет получено в вызове [GetMessage](#) . Так как в этом примере используется **MOD_ALT** со значением **MOD_NOREPEAT** для *fsModifiers*, поток получит другое сообщение **WM_HOTKEY** только при освобождении клавиши "b", а затем снова нажатой клавиши ALT.

C++

```
#include "stdafx.h"

int _cdecl _tmain (
    int argc,
    TCHAR *argv[])
{
    if (RegisterHotKey(
        NULL,
```

```

    1,
    MOD_ALT | MOD_NOREPEAT,
    0x42)) //0x42 is 'b'
{
    _tprintf(_T("Hotkey 'ALT+b' registered, using MOD_NOREPEAT
flag\n"));
}

MSG msg = {0};
while (GetMessage(&msg, NULL, 0, 0) != 0)
{
    if (msg.message == WM_HOTKEY)
    {
        _tprintf(_T("WM_HOTKEY received\n"));
    }
}

return 0;
}

```

Требования

Минимальная версия клиента	Windows Vista [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

Основные понятия

[GlobalAddAtom](#)

[Ввод с клавиатуры](#)

Справочные материалы

[Регистрация горячего ключа для текущего приложения \(CSRegisterHotkey\)](#)

[Регистрация горячего ключа для текущего приложения \(CppRegisterHotkey\)](#)

[Регистрация горячего ключа для текущего приложения \(VBRegisterHotkey\)](#)

Примеры

[UnregisterHotKey](#)

[WM_HOTKEY](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция RegisterRawInputDevices (winuser.h)

Статья 27.08.2023

Регистрирует устройства, которые предоставляют необработанные входные данные.

Синтаксис

C++

```
BOOL RegisterRawInputDevices(
    [in] PCRAWINPUTDEVICE pRawInputDevices,
    [in] UINT             uiNumDevices,
    [in] UINT             cbSize
);
```

Параметры

[in] pRawInputDevices

Тип: PCRAWINPUTDEVICE

Массив структур [RAWINPUTDEVICE](#), представляющих устройства, которые предоставляют необработанные входные данные.

[in] uiNumDevices

Тип: UINT

Число структур [RAWINPUTDEVICE](#), на которые указывает *pRawInputDevices*.

[in] cbSize

Тип: UINT

Размер структуры [RAWINPUTDEVICE](#) в байтах.

Возвращаемое значение

Тип: BOOL

Значение **TRUE**, если функция выполнена успешно; в противном случае — **FALSE**.

Если функция завершается сбоем, вызовите [Метод GetLastError для получения дополнительных сведений](#).

Комментарии

Для получения [WM_INPUT](#) сообщений приложение должно сначала зарегистрировать необработанные устройства ввода с помощью [RegisterRawInputDevices](#). По умолчанию приложение не получает необработанные входные данные.

Для получения [WM_INPUT_DEVICE_CHANGE](#) сообщений приложение должно указать флаг RIDEV_DEVNOTIFY для каждого класса устройства, заданного полями usUsagePage и usUsage структуры [RAWINPUTDEVICE](#). По умолчанию приложение не получает [WM_INPUT_DEVICE_CHANGE](#) уведомлений о поступлении и удалении необработанного устройства ввода.

Если в структуре [RAWINPUTDEVICE](#) установлен флаг RIDEV_REMOVE, а параметр hwndTarget не имеет значения NULL, проверка параметра завершится ошибкой.

Для получения необработанных входных данных в рамках процесса может быть зарегистрировано только одно окно для каждого класса необработанного устройства ввода (окно, переданное в последнем вызове [RegisterRawInputDevices](#)). По этой причине [RegisterRawInputDevices](#) не следует использовать из библиотеки, так как это может помешать любой логике обработки необработанных входных данных, уже присутствующей в приложениях, которые ее загружают.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

Набор API

ext-ms-win-ntuser-rawinput-l1-1-0 (представлено в Windows 10 версии 10.0.14393)

См. также раздел

Основные понятия

[RAWINPUTDEVICE](#)

[Необработанные входные данные](#)

Справочные материалы

[WM_INPUT](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция ReleaseCapture (winuser.h)

Статья 08.03.2023

Освобождает захват мыши из окна в текущем потоке и восстанавливает обычную обработку ввода мыши. Окно, которое захватило мышь, получает все входные данные мыши независимо от положения курсора, за исключением случаев нажатия кнопки мыши, пока горячая точка курсора находится в окне другого потока.

Синтаксис

C++

```
BOOL ReleaseCapture();
```

Возвращаемое значение

Тип: **BOOL**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Приложение вызывает эту функцию после вызова функции [SetCapture](#).

Примеры

Пример см. в разделе "[Рисование линий](#)" с помощью мыши.

Требования

Минимальная версия
клиента

Windows 2000 Professional [только классические
приложения]

Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-mouse-l1-1-0 (представлен в Windows 8)

См. также раздел

Основные понятия

[GetCapture](#)

[Ввод с помощью мыши](#)

Справочные материалы

[SetCapture](#)

[WM_CAPTURECHANGED](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

структура RID_DEVICE_INFO (winuser.h)

Статья 28.08.2023

Определяет необработанные входные данные, поступающие с любого устройства.

Синтаксис

C++

```
typedef struct tagRID_DEVICE_INFO {
    DWORD cbSize;
    DWORD dwType;
    union {
        RID_DEVICE_INFO_MOUSE     mouse;
        RID_DEVICE_INFO_KEYBOARD keyboard;
        RID_DEVICE_INFO_HID       hid;
    } DUMMYUNIONNAME;
} RID_DEVICE_INFO, *PRID_DEVICE_INFO, *LPRID_DEVICE_INFO;
```

Члены

`cbSize`

Тип: `DWORD`

Размер структуры `RID_DEVICE_INFO` в байтах.

`dwType`

Тип: `DWORD`

Тип необработанных входных данных. Этот элемент может быть одним из следующих значений.

Значение	Значение
<code>RIM_TYPEMOUSE</code> 0	Данные поступают с мыши.
<code>RIM_TYPEKEYBOARD</code> 1	Данные поступают с клавиатуры.
<code>RIM_TYPEHID</code> 2	Данные поступают из HID, который не является клавиатурой или мышью.

DUMMYUNIONNAME

DUMMYUNIONNAME.mouse

Тип: [RID_DEVICE_INFO_MOUSE](#)

Если dwType имеет RIM_TYPEMOUSE, это [RID_DEVICE_INFO_MOUSE](#) структура, которая определяет мышь.

DUMMYUNIONNAME.keyboard

Тип: [RID_DEVICE_INFO_KEYBOARD](#)

Если dwType имеет RIM_TYPEKEYBOARD, это [структура RID_DEVICE_INFO_KEYBOARD](#), которая определяет клавиатуру.

DUMMYUNIONNAME.hid

Тип: [RID_DEVICE_INFO_HID](#)

Если dwType имеет RIM_TYPEHID, это [RID_DEVICE_INFO_HID](#) структура, которая определяет устройство HID.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

Основные понятия

[GetRawInputDeviceInfo](#)

[RID_DEVICE_INFO_HID](#)

[RID_DEVICE_INFO_KEYBOARD](#)

[RID_DEVICE_INFO_MOUSE](#)

[Необработанные входные данные](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

структура RID_DEVICE_INFO_HID (winuser.h)

Статья 28.08.2023

Определяет необработанные входные данные, поступающие от указанного устройства HID.

Синтаксис

C++

```
typedef struct tagRID_DEVICE_INFO_HID {
    DWORD dwVendorId;
    DWORD dwProductId;
    DWORD dwVersionNumber;
    USHORT usUsagePage;
    USHORT usUsage;
} RID_DEVICE_INFO_HID, *PRID_DEVICE_INFO_HID;
```

Члены

`dwVendorId`

Тип: **DWORD**

Идентификатор поставщика для HID.

`dwProductId`

Тип: **DWORD**

Идентификатор продукта для HID.

`dwVersionNumber`

Тип: **DWORD**

Номер версии HID.

`usUsagePage`

Тип: **USHORT**

Страница использования коллекции верхнего уровня для устройства.

usUsage

Тип: USHORT

Использование коллекции верхнего уровня для устройства.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

[Основные понятия](#)

[RID_DEVICE_INFO](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

структура RID_DEVICE_INFO_KEYBOARD (winuser.h)

Статья 28.08.2023

Определяет необработанные входные данные, поступающие с указанной клавиатурой.

Синтаксис

C++

```
typedef struct tagRID_DEVICE_INFO_KEYBOARD {
    DWORD dwType;
    DWORD dwSubType;
    DWORD dwKeyboardMode;
    DWORD dwNumberOfFunctionKeys;
    DWORD dwNumberOfIndicators;
    DWORD dwNumberOfKeysTotal;
} RID_DEVICE_INFO_KEYBOARD, *PRID_DEVICE_INFO_KEYBOARD;
```

Члены

`dwType`

Тип: **DWORD**

Тип клавиатуры. См. [Примечания](#).

Значение	Описание
0x4	Улучшенные клавиатуры с 101 или 102 клавишами (и совместимые)
0x7	Японская клавиатура
0x8	Корейская клавиатура
0x51	Неизвестный тип или клавиатура HID

`dwSubType`

Тип: **DWORD**

Подтип клавиатуры, зависящий от поставщика. См. [Примечания](#).

`dwKeyboardMode`

Тип: **DWORD**

Режим кода сканирования. Обычно 1, что означает, что используется *набор кода сканирования 1*. См. [раздел Спецификация кода сканирования клавиатуры](#).

`dwNumberOfFunctionKeys`

Тип: **DWORD**

Количество функциональных клавиш на клавиатуре.

`dwNumberOfIndicators`

Тип: **DWORD**

Количество светодиодных индикаторов на клавиатуре.

`dwNumberOfKeysTotal`

Тип: **DWORD**

Общее количество клавиш на клавиатуре.

Комментарии

Сведения о типах клавиатуры, подтипах, режимах кода сканирования и связанных раскладках клавиатуры см. в документации по заголовкам *kbd.h*, *ntdd8042.h* и *ntddkbd.h* в windows SDK, а также в [примерах раскладки клавиатуры](#).

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

[Основные понятия](#)

[RID_DEVICE_INFO](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

[структура KEYBOARD_ATTRIBUTES](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

структура RID_DEVICE_INFO_MOUSE (winuser.h)

Статья 28.08.2023

Определяет необработанные входные данные, поступающие от указанной мыши.

Синтаксис

C++

```
typedef struct tagRID_DEVICE_INFO_MOUSE {
    DWORD dwId;
    DWORD dwNumberOfButtons;
    DWORD dwSampleRate;
    BOOL fHasHorizontalWheel;
} RID_DEVICE_INFO_MOUSE, *PRID_DEVICE_INFO_MOUSE;
```

Члены

`dwId`

Тип: **DWORD**

Битовое поле свойств идентификации устройства мыши:

Значение	Константа ntddmou.h	Описание
0x0080	MOUSE_HID_HARDWARE	Мышь HID
0x0100	WHEELMOUSE_HID_HARDWARE	Мышь hid wheel mouse
0x8000	HORIZONTAL_WHEEL_PRESENT	Мышь с горизонтальным колесиком

`dwNumberOfButtons`

Тип: **DWORD**

Число кнопок для мыши.

`dwSampleRate`

Тип: **DWORD**

Количество точек данных в секунду. Эти сведения могут быть неприменимы для всех устройств с мышью.

fHasHorizontalWheel

Тип: **BOOL**

ЗНАЧЕНИЕ TRUE, если мышь имеет колесико для горизонтальной прокрутки; в противном случае — **FALSE**.

Windows XP: Этот член поддерживается только начиная с Windows Vista.

Комментарии

Для мыши страница "Использование" имеет значение 1, а значение "Использование" — 2.

Требования

Минимальная версия клиента	Windows XP [только классические приложения]
Минимальная версия сервера	Windows Server 2003 [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

[Основные понятия](#)

[RID_DEVICE_INFO](#)

[Необработанные входные данные](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция SendInput (winuser.h)

Статья 28.08.2023

Синтезирует нажатия клавиш, движения мыши и нажатия кнопок.

Синтаксис

C++

```
UINT SendInput(  
    [in] UINT    cInputs,  
    [in] LPINPUT pInputs,  
    [in] int     cbSize  
) ;
```

Параметры

[in] cInputs

Тип: **UINT**

Количество структур в массиве *pInputs* .

[in] pInputs

Тип: **LPINPUT**

Массив структур **INPUT** . Каждая структура представляет событие для вставки в поток ввода с клавиатуры или мыши.

[in] cbSize

Тип: **int**

Размер структуры **INPUT** в байтах. Если *cbSize* не соответствует размеру структуры **INPUT** , функция завершается ошибкой.

Возвращаемое значение

Тип: **UINT**

Функция возвращает количество событий, успешно вставляемых в поток ввода с клавиатуры или мыши. Если функция возвращает ноль, входные данные уже заблокированы другим потоком. Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Эта функция завершается сбоем, если она заблокирована UIPI. Обратите внимание, что ни [GetLastError](#), ни возвращаемое значение не указывают, что сбой был вызван блокировкой UIPI.

Комментарии

Эта функция зависит от UIPI. Приложениям разрешено внедрять входные данные только в приложения с равным или меньшим уровнем целостности.

Функция [SendInput](#) вставляет события в структурах [INPUT](#) последовательно в поток ввода с клавиатуры или мыши. Эти события не чередуются с другими событиями ввода с помощью клавиатуры или мыши, вставленными пользователем (с помощью клавиатуры или мыши) или вызовами [keybd_event](#), [mouse_event](#) или другими вызовами [SendInput](#).

Эта функция не сбрасывает текущее состояние клавиатуры. Все клавиши, которые уже нажаты при вызове функции, могут мешать событиям, создаваемым этой функцией. Чтобы избежать этой проблемы, проверка состояния клавиатуры с помощью функции [GetAsyncKeyState](#) и исправьте при необходимости.

Так как сенсорная клавиатура использует суррогатные макросы, определенные в `winnls.h`, для отправки входных данных в систему, прослушиватель на перехватчике событий клавиатуры должен декодировать входные данные, поступающие от сенсорной клавиатуры. Дополнительные сведения см. в разделе [Суррогаты и дополнительные символы](#).

Приложение со специальными возможностями может использовать [SendInput](#) для внедрения нажатий клавиш, соответствующих сочетаниям клавиш запуска приложения, которые обрабатываются оболочкой. Эта функция не гарантирует работу с другими типами приложений.

Пример

C++

```
*****  
//  
// Sends Win + D to toggle to the desktop
```

```

// ****
void ShowDesktop()
{
    OutputString(L"Sending 'Win-D'\r\n");
    INPUT inputs[4] = {};
    ZeroMemory(inputs, sizeof(inputs));

    inputs[0].type = INPUT_KEYBOARD;
    inputs[0].ki.wVk = VK_LWIN;

    inputs[1].type = INPUT_KEYBOARD;
    inputs[1].ki.wVk = 'D';

    inputs[2].type = INPUT_KEYBOARD;
    inputs[2].ki.wVk = 'D';
    inputs[2].ki.dwFlags = KEYEVENTF_KEYUP;

    inputs[3].type = INPUT_KEYBOARD;
    inputs[3].ki.wVk = VK_LWIN;
    inputs[3].ki.dwFlags = KEYEVENTF_KEYUP;

    UINT uSent = SendInput(ARRAYSIZE(inputs), inputs, sizeof(INPUT));
    if (uSent != ARRAYSIZE(inputs))
    {
        OutputString(L"SendInput failed: 0x%x\n",
        HRESULT_FROM_WIN32(GetLastError()));
    }
}

```

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

Основные понятия

[GetAsyncKeyState](#)

[INPUT](#)

[Ввод с клавиатуры](#)

[Справочные материалы](#)

[Суррогаты и дополнительные символы](#)

[keybd_event](#)

[mouse_event](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да  Нет

[Получить справку в Microsoft Q&A](#)

Функция SetActiveWindow (winuser.h)

Статья 28.08.2023

Активирует окно. Окно должно быть присоединено к очереди сообщений вызывающего потока.

Синтаксис

C++

```
HWND SetActiveWindow(  
    [in] HWND hWnd  
>;
```

Параметры

[in] hWnd

Тип: HWND

Дескриптор для запускаемого окна верхнего уровня.

Возвращаемое значение

Тип: HWND

Если функция выполняется успешно, возвращается дескриптор для ранее активного окна.

Если функция завершается сбоем, возвращается значение NULL. Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Функция SetActiveWindow активирует окно, но не активирует, если приложение находится в фоновом режиме. Окно будет выведено на передний план (в верхней части [Z-порядка](#)), если его приложение находится на переднем плане, когда система активирует окно.

Если окно, определенное параметром *hWnd*, было создано вызывающим потоком, то для состояния активного окна вызывающего потока устанавливается значение *hWnd*. В противном случае для состояния активного окна вызывающего потока устанавливается значение **NULL**.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-window-l1-1-4 (появилась в Windows 10 версии 10.0.14393)

См. также раздел

Основные понятия

[GetActiveWindow](#)

[Ввод с клавиатуры](#)

Справочные материалы

[SetForegroundWindow](#)

[WM_ACTIVATE](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция SetCapture (winuser.h)

Статья 28.08.2023

Задает для захвата мыши указанное окно, принадлежащее текущему потоку.

SetCapture захватывает входные данные мыши, когда указатель мыши находится над окном захвата или когда кнопка мыши была нажата, когда мышь находилась над окном захвата, а кнопка все еще не работает. Только одно окно за раз может захватывать мышь.

Если курсор мыши находится над окном, созданным другим потоком, система будет направлять ввод мыши в указанное окно только в том случае, если кнопка мыши не работает.

Синтаксис

C++

```
HWND SetCapture(  
    [in] HWND hWnd  
>;
```

Параметры

[in] hWnd

Тип: **HWND**

Дескриптор для окна в текущем потоке для захвата мыши.

Возвращаемое значение

Тип: **HWND**

Возвращаемое значение — это дескриптор для окна, которое ранее захватило мышь. Если такого окна нет, возвращается значение **NULL**.

Комментарии

Захватывать мышь может только окно переднего плана. Когда фоновое окно пытается сделать это, окно получает сообщения только для событий мыши,

которые возникают, когда курсор находится в видимой части окна. Кроме того, даже если окно переднего плана захватило мышь, пользователь по-прежнему может щелкнуть другое окно и вывести его на передний план.

Если окно больше не требует ввода всех данных с помощью мыши, поток, создавший окно, должен вызвать функцию [ReleaseCapture](#), чтобы освободить мышь.

Эту функцию нельзя использовать для записи входных данных с помощью мыши, предназначенных для другого процесса.

При захвате мыши клавиши меню и другие ускорители клавиатуры не работают.

Примеры

Пример см. в разделе [Рисование линий с помощью мыши](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-mouse-l1-1-0 (появилось в Windows 8)

См. также раздел

[Основные понятия](#)

[GetCapture](#)

[Ввод с помощью мыши](#)

Справочные материалы

[ReleaseCapture](#)

[WM_CAPTURECHANGED](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция SetDoubleClickTime (winuser.h)

Статья 28.08.2023

Задает время двойного щелчка мыши. Двойной щелчок — это последовательность из двух щелчков кнопки мыши, второй из них происходит в течение указанного времени после первого. Время двойного щелчка — это максимальное количество миллисекунд, которое может произойти между первым и вторым щелчком при двойном щелчке.

Синтаксис

C++

```
BOOL SetDoubleClickTime(  
    [in] UINT unnamedParam1  
);
```

Параметры

[in] unnamedParam1

Тип: **UINT**

Количество миллисекунд, которое может произойти между первым и вторым щелчком двойного щелчка. Если этот параметр имеет значение 0, система использует время двойного щелчка по умолчанию, равное 500 миллисекундам. Если значение этого параметра превышает 5000 миллисекунда, система устанавливает значение 5000 миллисекундах.

Возвращаемое значение

Тип: **BOOL**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Функция `SetDoubleClickTime` изменяет время двойного щелчка для всех окон в системе.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-mouse-l1-1-1 (представлено в Windows 10 версии 10.0.14393)

См. также раздел

[Основные понятия](#)

[GetDoubleClickTime](#)

[Ввод с помощью мыши](#)

[Справочные материалы](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да	 Нет
--	--

[Получить справку в Microsoft Q&A](#)

Функция SetFocus (winuser.h)

Статья 28.08.2023

Устанавливает фокус клавиатуры в указанное окно. Окно должно быть присоединено к очереди сообщений вызывающего потока.

Синтаксис

C++

```
HWND SetFocus(  
    [in, optional] HWND hWnd  
);
```

Параметры

[in, optional] hWnd

Тип: HWND

Дескриптор окна, которое получит ввод с клавиатуры. Если этот параметр имеет значение NULL, нажатия клавиш игнорируются.

Возвращаемое значение

Тип: HWND

Если функция выполнена успешно, возвращается дескриптор для окна, в которое ранее был фокус клавиатуры. Если параметр *hWnd* недопустим или окно не подключено к очереди сообщений вызывающего потока, возвращаемое значение равно NULL. Чтобы получить расширенные сведения об ошибке, вызовите функцию [GetLastError](#).

Расширенное ERROR_INVALID_PARAMETER ошибок (0x57) означает, что окно находится в отключенном состоянии.

Комментарии

Эта функция отправляет [WM_KILLFOCUS](#) сообщение в окно, которое теряет фокус клавиатуры, и [сообщение WM_SETFOCUS](#) в окно, которое получает фокус

клавиатуры. Он также активирует либо окно, которое получает фокус, либо родительское окно, которое получает фокус.

Если окно активно, но не имеет фокуса, любая нажатая клавиша создает сообщение [WM_SYSCHAR](#), [WM_SYSKEYDOWN](#) или [WM_SYSKEYUP](#). Если клавиша VK_MENU также нажата, задается бит 30 параметра *lParam* сообщения. В противном случае для создаваемых сообщений этот бит не задан.

С помощью функции [AttachThreadInput](#) поток может присоединить свою обработку входных данных к другому потоку. Это позволяет потоку вызывать SetFocus, чтобы установить фокус клавиатуры на окно, присоединенное к очереди сообщений другого потока.

Примеры

Пример см. в разделе [Инициализация диалогового окна](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll
Набор API	ext-ms-win-ntuser-window-l1-1-4 (появилась в Windows 10 версии 10.0.14393)

См. также раздел

Функция [AttachThreadInput](#), функция [GetFocus](#), [WM_KILLFOCUS](#), [WM_SETFOCUS](#), [WM_SYSCHAR](#), [WM_SYSKEYDOWN](#), [WM_SYSKEYUP](#), ввод с клавиатуры

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция SetKeyboardState (winuser.h)

Статья 28.08.2023

Копирует массив состояний клавиш клавиатуры в таблицу входных данных клавиатуры вызывающего потока. Это та же таблица, к которым обращаются функции [GetKeyboardState](#) и [GetKeyState](#). Изменения, внесенные в эту таблицу, не влияют на ввод с клавиатуры в любой другой поток.

Синтаксис

C++

```
BOOL SetKeyboardState(  
    [in] LPBYTE lpKeyState  
>;
```

Параметры

[in] lpKeyState

Тип: **LPBYTE**

Указатель на 256-байтовой массив, содержащий состояния клавиши клавиатуры.

Возвращаемое значение

Тип: **BOOL**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Поскольку функция **SetKeyboardState** изменяет входное состояние вызывающего потока, а не глобальное состояние входных данных системы, приложение не может использовать **SetKeyboardState** для установки индикаторов NUM LOCK, CAPS LOCK или SCROLL LOCK (или японского KANA) на клавиатуре. Их можно задать или очистить с помощью [SendInput](#) для имитации нажатий клавиш.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [GetSystemMetrics](#)
- [MapVirtualKey](#)
- [SetKeyboardState](#)
- [SendInput](#)
- [keybd_event](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция SwapMouseButton (winuser.h)

Статья 28.08.2023

Изменяет или восстанавливает значение левой и правой кнопок мыши.

Синтаксис

C++

```
BOOL SwapMouseButton(  
    [in] BOOL fSwap  
);
```

Параметры

[in] fSwap

Тип: **BOOL**

Если этот параметр имеет значение **TRUE**, левая кнопка создает сообщения правой кнопки, а правая — сообщения левой кнопки. Если этот параметр имеет значение **FALSE**, кнопки восстанавливаются до исходного значения.

Возвращаемое значение

Тип: **BOOL**

Если значение кнопок мыши было отменено ранее, то до вызова функции возвращаемое значение будет ненулевым.

Если значение кнопок мыши не было изменено, возвращаемое значение равно нулю.

Комментарии

Переключение кнопок предоставляется для удобства для пользователей, которые используют мышь с левой рукой. Функция **SwapMouseButton** обычно вызывается только панель управления. Хотя приложение может свободно вызывать функцию, мышь является общим ресурсом, и изменение значения ее кнопок влияет на все приложения.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с помощью мыши](#)

[Справочные материалы](#)

[SetDoubleClickTime](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция ToAscii (winuser.h)

Статья 20.05.2023

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующий символ или символы. Функция преобразует код с помощью языка ввода и физической раскладки клавиатуры, определяемой дескриптором раскладки клавиатуры.

Чтобы указать дескриптор раскладки клавиатуры для перевода указанного кода, используйте функцию [ToAsciiEx](#).

ⓘ Примечание

Этот метод может неправильно работать с некоторыми **раскладками клавиатуры**, которые могут создавать несколько символов (т. е. лигатуры) и / или дополнительные символы Юникода при одном нажатии клавиши. Настоятельно рекомендуется использовать **методы ToUnicode** или **ToUnicodeEx**, которые правильно обрабатывают такие случаи.

Синтаксис

C++

```
int ToAscii(
    [in]          UINT      uVirtKey,
    [in]          UINT      uScanCode,
    [in, optional] const BYTE *lpKeyState,
    [out]         LPWORD    lpChar,
    [in]          UINT      uFlags
);
```

Параметры

[in] uVirtKey

Тип: **UINT**

Код виртуального ключа для перевода. См . [раздел Коды виртуальных ключей](#).

[in] uScanCode

Тип: **UINT**

Код аппаратного сканирования ключа для преобразования. Бит высокого порядка этого значения устанавливается, если клавиша находится вверх (не нажата).

[in, optional] `lpKeyState`

Тип: **const BYTE***

Указатель на 256-байтовый массив, содержащий текущее состояние клавиатуры. Каждый элемент (байт) в массиве содержит состояние одного ключа. Если задан бит байта высокого порядка, клавиша опускается (нажата).

Низкий бит, если он задан, указывает, что ключ включен. В этой функции имеет значение только бит переключателя клавиши CAPS LOCK. Состояние переключателя клавиш NUM LOCK и SCROLL LOCK игнорируется.

[out] `lpChar`

Тип: **LPWORD**

Буфер, получающий переведенный символ или символы. Байт нижнего порядка содержит первый символ, а байт высокого порядка — второй символ, если он имеется.

[in] `uFlags`

Тип: **UINT**

Этот параметр должен иметь значение 1, если меню активно, или 0 в противном случае.

Возвращаемое значение

Тип: **int**

Возвращаемое значение является одним из следующих значений.

Возвращаемое значение	Описание
0	Указанная виртуальная клавиша не имеет перевода текущего состояния клавиатуры.
1	Один символ был скопирован в буфер.
2	В буфер были скопированы два символа. Обычно это происходит, когда неотключаемый символ

(диакритический или диакритический символ), хранящийся в раскладке клавиатуры, не может быть составлен с указанной виртуальной клавишей для формирования одного символа.

Комментарии

Параметров, предоставленных функции `ToAscii`, может быть недостаточно для преобразования кода виртуальной клавиши, так как предыдущая неактивная клавиша хранится в раскладке клавиатуры.

Как правило, `ToAscii` выполняет преобразование на основе кода виртуального ключа. Однако в некоторых случаях бит 15 параметра `uScanCode` можно использовать для различия нажатия клавиши и выпуска клавиши. Код сканирования используется для перевода сочетаний клавиш `ALT+ числовой клавиши`.

Хотя `NUM LOCK` является выключателем, влияющим на поведение клавиатуры, `ToAscii` игнорирует параметр переключателя (низкий бит) `lpKeyState` (`VK_NUMLOCK`), так как только параметра `uVirtKey` достаточно, чтобы отличать клавиши перемещения курсора (`VK_HOME`, `VK_INSERT` и т. д.) от числовых клавиш (`VK_DECIMAL`, `VK_NUMPAD0` - `VK_NUMPAD9`).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с клавиатуры](#)

[OemKeyScan](#)

[Справочные материалы](#)

[ToAsciiEx](#)

[ToUnicode](#)

[VkKeyScan](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция ToAsciiEx (winuser.h)

Статья 20.05.2023

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы или символы. Функция преобразует код с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода.

ⓘ Примечание

Этот метод может неправильно работать с некоторыми **раскладками клавиатуры**, которые могут создавать несколько символов (например, лигатуры) и (или) дополнительные символы Юникода при нажатии одной клавиши. Настоятельно рекомендуется использовать методы **ToUnicode** или **ToUnicodeEx**, которые правильно обрабатывают такие случаи.

Синтаксис

C++

```
int ToAsciiEx(
    [in]          UINT      uVirtKey,
    [in]          UINT      uScanCode,
    [in, optional] const BYTE *lpKeyState,
    [out]         LPWORD    lpChar,
    [in]          UINT      uFlags,
    [in, optional] HKL      dwhkl
);
```

Параметры

[in] uVirtKey

Тип: **UINT**

Код виртуального ключа для преобразования. См . [раздел Коды виртуальных ключей](#).

[in] uScanCode

Тип: **UINT**

Код аппаратного сканирования ключа для преобразования. Бит высокого порядка этого значения задается, если клавиша вверх (не нажата).

[in, optional] lpKeyState

Тип: const BYTE*

Указатель на 256-байтовый массив, содержащий текущее состояние клавиатуры. Каждый элемент (байт) в массиве содержит состояние одного ключа. Если задан бит высокого порядка байта, клавиша опускается (нажата).

Низкий бит, если он задан, указывает, что ключ включен. В этой функции имеет значение только бит переключателя клавиши CAPS LOCK. Состояние переключателя клавиш NUM LOCK и SCOLL LOCK игнорируется.

[out] lpChar

Тип: LPWORD

Указатель на буфер, который получает переведенный символ или символы. Байт нижнего порядка содержит первый символ, а байт высокого порядка — второй, если он имеется.

[in] uFlags

Тип: UINT

Этот параметр должен иметь значение 1, если меню активно, в противном случае — ноль.

[in, optional] dwhkl

Тип: HKL

Входной идентификатор языкового стандарта, используемый для перевода кода. Этот параметр может быть любым идентификатором входного языкового стандарта, ранее возвращенным функцией [LoadKeyboardLayout](#).

Возвращаемое значение

Тип: int

Возвращаемое значение является одним из следующих значений.

Возвращаемое значение	Описание

0	Указанная виртуальная клавиша не имеет перевода текущего состояния клавиатуры.
1	Один символ был скопирован в буфер.
2	В буфер были скопированы два символа. Обычно это происходит, когда символ недоставленной клавиши (акцентный или диакритический), хранящийся в раскладке клавиатуры, не может быть составлен с помощью указанной виртуальной клавиши для формирования одного символа.

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Параметров, предоставленных функции `ToAsciiEx`, может быть недостаточно для преобразования кода виртуальной клавиши, так как предыдущая недопустимая клавиша хранится в раскладке клавиатуры.

Как правило, `ToAsciiEx` выполняет преобразование на основе кода виртуального ключа. Однако в некоторых случаях для различения нажатия клавиши и отпускания клавиш можно использовать бит 15 параметра `uScanCode`. Код сканирования используется для перевода сочетаний клавиш ALT+числовой клавиши.

Хотя NUM LOCK является переключателем, влияющим на поведение клавиатуры, `ToAsciiEx` игнорирует параметр переключателя (низкий бит) `lpKeyState` (`VK_NUMLOCK`), так как только параметра `uVirtKey` достаточно, чтобы отличать клавиши перемещения курсора (`VK_HOME`, `VK_INSERT` и т. д.) от числовых клавиш (`VK_DECIMAL`, `VK_NUMPAD0` - `VK_NUMPAD9`).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows

Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

Основные понятия

[Ввод с клавиатуры](#)

[LoadKeyboardLayout](#)

[MapVirtualKeyEx](#)

Справочные материалы

[ToUnicodeEx](#)

[VkKeyScan](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

перечисление TOOLTIP_DISMISS_FLAGS (winuser.h)

Статья 15.08.2023

Определяет константы, указывающие, зарегистрировано или отменено окно для получения уведомлений о закрытии всплывающих подсказок.

Синтаксис

C++

```
typedef enum {
    TDF_REGISTER,
    TDF_UNREGISTER
} TOOLTIP_DISMISS_FLAGS;
```

Константы

TDF_REGISTER

Окно зарегистрировано для получения уведомлений об закрытии подсказки.

TDF_UNREGISTER

Окно не зарегистрировано от получения уведомлений о закрытии подсказки.

Комментарии

Это перечисление используется функцией [RegisterForTooltipDismissNotification](#).

Требования

Минимальная версия клиента

сборка Windows 11 22621

Верхняя часть

winuser.h

См. также раздел

[RegisterForTooltipDismissNotification](#), подсказка

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция ToUnicode (winuser.h)

Статья 28.08.2023

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы Юникода.

Синтаксис

C++

```
int ToUnicode(
    [in]          UINT      wVirtKey,
    [in]          UINT      wScanCode,
    [in, optional] const BYTE *lpKeyState,
    [out]         LPWSTR    pwszBuff,
    [in]          int       cchBuff,
    [in]          UINT      wFlags
);
```

Параметры

[in] wVirtKey

Тип: **UINT**

Код виртуального ключа для преобразования. См . [раздел Коды виртуальных ключей](#).

[in] wScanCode

Тип: **UINT**

[Код аппаратного сканирования](#) ключа для преобразования. Бит высокого порядка этого значения задается, если ключ включен.

[in, optional] lpKeyState

Тип: **const BYTE***

Указатель на 256-байтовый массив, содержащий текущее состояние клавиатуры. Каждый элемент (байт) в массиве содержит состояние одного ключа.

Если задан бит высокого порядка байта, ключ не работает. Низкий бит, если он задан, указывает, что ключ включен. В этой функции имеет значение только бит переключателя клавиши CAPS LOCK. Состояние переключателя клавиш NUM LOCK и SCROLL LOCK игнорируется. Дополнительные сведения см. в разделе [GetKeyboardState](#).

[out] pwszBuff

Тип: LPWSTR

Буфер, который получает переведенные символы или символы в виде массива единиц кода UTF-16. Этот буфер может быть возвращен без завершения со значением NULL, даже если имя переменной предполагает, что он завершается со значением NULL. Вы можете использовать возвращаемое значение этого метода, чтобы определить, сколько символов было записано.

[in] cchBuff

Тип: int

Размер (в символах) буфера, на который указывает параметр *pwszBuff*.

[in] wFlags

Тип: UINT

Поведение функции.

Если задан бит 0, меню активно. В этом режиме **сочетания клавиш ALT+числовой клавиатуры** не обрабатываются.

Если задан бит 2, состояние клавиатуры не изменяется (Windows 10 версии 1607 и более поздние версии)

Все остальные биты (до 31) зарезервированы.

Возвращаемое значение

Тип: int

Функция возвращает одно из следующих значений.

Возвращаемое значение	Описание
значение < 0	Указанная виртуальная клавиша является символом мертвых клавиши (диакритической или

диакритической). Это значение возвращается независимо от раскладки клавиатуры, даже если было введено несколько символов, которые хранятся в состоянии клавиатуры. Если это возможно, даже при использовании раскладок клавиатуры в Юникоде функция записала в буфер, заданный *pwszBuff*, версию интервала символа недоставленной клавиши.

Например, функция записывает символ ACUTE ACCENT (U+00B4), а не символ COMBINING ACUTE ACCENT (U+0301).

0	Указанная виртуальная клавиша не имеет перевода текущего состояния клавиатуры. В буфер, указанный <i>pwszBuff</i> , ничего не записано.
---	---

значение > 0	Одна или несколько единиц кода UTF-16 были записаны в буфер, указанный <i>pwszBuff</i> . Возвращаемый <i>pwszBuff</i> может содержать больше символов, чем указано в возвращаемом значении. В этом случае все дополнительные символы недопустимы и должны игнорироваться.
--------------	---

Комментарии

Чтобы указать дескриптор раскладки клавиатуры для перевода указанного кода, используйте функцию [ToUnicodeEx](#).

Некоторые раскладки клавиатуры могут возвращать несколько символов или дополнительных символов в качестве [суррогатных пар](#) в *pwszBuff*. Если не удалось объединить неактивный символ (акцентный или диакритический символ), хранящийся в раскладке клавиатуры, с указанной виртуальной клавишей для формирования одного символа, то предыдущий введенный мертвый символ можно объединить с текущим символом.

Параметров, предоставленных функции [ToUnicodeEx](#), может быть недостаточно для преобразования кода виртуальной клавиши, так как предыдущая [недопустимая клавиша](#) хранится в раскладке клавиатуры.

Как правило, [ToUnicode](#) выполняет преобразование на основе кода виртуального ключа. Однако в некоторых случаях бит 15 параметра *wScanCode* можно использовать для различия нажатия клавиши и ее отпускания (например, для записи клавиш ALT+numpad).

Так как [ToUnicode](#) преобразует код виртуальной клавиши, он также изменяет состояние буфера клавиатуры в режиме ядра. Это изменение состояния влияет на

мертвые ключи, лигатуры, **alt+числовую** клавиатуру и т. д. Это также может вызвать нежелательные побочные эффекты при использовании в сочетании с [TranslateMessage](#) (что также изменяет состояние буфера клавиатуры в режиме ядра).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с клавиатуры](#)

[Справочные материалы](#)

[ToAscii](#)

[ToUnicodeEx](#)

[VkKeyScan](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция ToUnicodeEx (winuser.h)

Статья 14.04.2023

Преобразует указанный код виртуальной клавиши и состояние клавиатуры в соответствующие символы Юникода.

Синтаксис

C++

```
int ToUnicodeEx(
    [in]          UINT      wVirtKey,
    [in]          UINT      wScanCode,
    [in]          const BYTE *lpKeyState,
    [out]         LPWSTR    pwszBuff,
    [in]          int       cchBuff,
    [in]          UINT      wFlags,
    [in, optional] HKL      dwhkl
);
```

Параметры

[in] wVirtKey

Тип: **UINT**

Код виртуального ключа, который необходимо преобразовать. См . [раздел Коды виртуальных ключей](#).

[in] wScanCode

Тип: **UINT**

[Код аппаратного сканирования](#) ключа для преобразования. Бит высокого порядка этого значения задается, если ключ включен.

[in] lpKeyState

Тип: **const BYTE***

Указатель на 256-байтовый массив, содержащий текущее состояние клавиатуры. Каждый элемент (байт) в массиве содержит состояние одного ключа.

Если задан бит высокого порядка байта, ключ не работает. Низкий бит, если он задан, указывает, что ключ включен. В этой функции имеет значение только бит переключателя клавиши CAPS LOCK. Состояние переключателя клавиш NUM LOCK и SCROLL LOCK игнорируется. Дополнительные сведения см. в разделе [GetKeyboardState](#).

[out] pwszBuff

Тип: LPWSTR

Буфер, который получает переведенные символы или символы в виде массива единиц кода UTF-16. Этот буфер может быть возвращен без завершения со значением NULL, даже если имя переменной предполагает, что он завершается со значением NULL. Вы можете использовать возвращаемое значение этого метода, чтобы определить, сколько символов было записано.

[in] cchBuff

Тип: int

Размер (в символах) буфера, на который указывает параметр *pwszBuff*.

[in] wFlags

Тип: UINT

Поведение функции.

Если задан бит 0, меню активно. В этом режиме **сочетания клавиш ALT+числовой клавиатуры** не обрабатываются.

Если задан бит 1, *ToUnicodeEx* будет переводить коды сканирования, помеченные как события разрыва ключа, в дополнение к обычной обработке событий создания ключей.

Если задан бит 2, состояние клавиатуры не изменяется (Windows 10 версии 1607 и более поздних версий).

Все остальные биты (до 31) зарезервированы.

[in, optional] dwhkl

Тип: HKL

Идентификатор входного языкового стандарта, используемый для перевода указанного кода. Этот параметр может быть любым идентификатором входного

языкового стандарта, ранее возвращенным функцией [LoadKeyboardLayout](#) .

Возвращаемое значение

Тип: `int`

Функция возвращает одно из следующих значений.

Возвращаемое значение	Описание
<code>значение < 0</code>	Указанная виртуальная клавиша является символом мертвых клавиши (диакритической или диакритической). Это значение возвращается независимо от раскладки клавиатуры, даже если несколько символов были введены и сохранены в состоянии клавиатуры. Если это возможно, даже при использовании раскладок клавиатуры в Юникоде функция записала в буфер, заданный <code>pwszBuff</code> , версию интервала символа недоставленного ключа. Например, функция записывает символ ACUTE ACCENT (U+00B4), а не символ COMBINING ACUTE ACCENT (U+0301).
0	Указанная виртуальная клавиша не имеет перевода текущего состояния клавиатуры. В буфер, указанный <code>pwszBuff</code> , ничего не записано.
<code>значение > 0</code>	Одна или несколько единиц кода UTF-16 были записаны в буфер, заданный <code>pwszBuff</code> . Возвращаемый <code>pwszBuff</code> может содержать больше символов, чем указано в возвращаемом значении. В этом случае все дополнительные символы недопустимы и должны игнорироваться.

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Некоторые раскладки клавиатуры могут возвращать несколько символов или дополнительных символов в качестве [суррогатных пар](#) в `pwszBuff`. Если в раскладке клавиатуры не удалось объединить с указанной виртуальной клавишей для формирования одного символа, то предыдущий введенный мертвый символ можно объединить с текущим символом.

Параметров, предоставленных функции `ToUnicodeEx`, может быть недостаточно для преобразования кода виртуальной клавиши, так как предыдущая [недопустимая клавиша](#) хранится в раскладке клавиатуры.

Как правило, `ToUnicodeEx` выполняет преобразование на основе кода виртуального ключа. Однако в некоторых случаях бит 15 параметра `wScanCode` можно использовать для различения нажатия клавиши и ее отпускания (например, для записи клавиш `ALT+numpad`).

Так как `ToUnicodeEx` преобразует код виртуального ключа, он также изменяет состояние буфера клавиатуры в режиме ядра. Это изменение состояния влияет на мертвые ключи, лигатуры, `alt+числовую` клавиатуру и т. д. Это также может вызвать нежелательные побочные эффекты при использовании в сочетании с `TranslateMessage` (что также изменяет состояние буфера клавиатуры в режиме ядра).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с клавиатуры](#)

[LoadKeyboardLayout](#)

[Справочные материалы](#)

[ToAsciiEx](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция TrackMouseEvent (winuser.h)

Статья 28.08.2023

Публикует сообщения, когда указатель мыши покидает окно или наносит указатель мыши на окно в течение указанного периода времени.

Примечание Функция `_TrackMouseEvent` вызывает `TrackMouseEvent`, если она существует, в противном случае `_TrackMouseEvent` эмулирует `TrackMouseEvent`.

Синтаксис

C++

```
BOOL TrackMouseEvent(  
    [in, out] LPTRACKMOUSEEVENT lpEventTrack  
);
```

Параметры

[in, out] `lpEventTrack`

Тип: `LPTRACKMOUSEEVENT`

Указатель на структуру `TRACKMOUSEEVENT`, содержащую сведения об отслеживании.

Возвращаемое значение

Тип: `BOOL`

Если функция выполнена успешно, возвращается ненулевое значение .

Если функция завершается сбоем, возвращаемое значение равно нулю.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Указатель мыши считается наведенным, если он остается в пределах указанного прямоугольника в течение указанного периода времени. Вызовите [SystemParametersInfo](#), и используют значения **SPI_GETMOUSEHOVERWIDTH**, **SPI_GETMOUSEHOVERHEIGHT** и **SPI_GETMOUSEHOVERTIME**, чтобы получить размер прямоугольника и время.

Функция может публиковать следующие сообщения.

Сообщение	Описание
WM_NCMOUSEOVER	То же значение, что и WM_MOUSEOVER за исключением этого, для неклиентной области окна.
WM_NCMOUSELEAVE	То же значение, что и WM_MOUSELEAVE за исключением этого, для неклиентной области окна.
WM_MOUSEOVER	Указатель мыши наведен на клиентную область окна в течение периода времени, указанного в предыдущем вызове TrackMouseEvent . Отслеживание при наведении указателя останавливается при создании этого сообщения. Приложение должно снова вызывать TrackMouseEvent , если требуется дальнейшее отслеживание поведения наведения указателя мыши.
WM_MOUSELEAVE	Мышь вышла из клиентской области окна, указанной в предыдущем вызове TrackMouseEvent . Все отслеживание, запрошенное TrackMouseEvent , отменяется при создании этого сообщения. Приложение должно вызывать TrackMouseEvent при повторном вводе окна мыши, если требуется дальнейшее отслеживание поведения наведения указателя мыши.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с помощью мыши](#)

[Другие ресурсы](#)

[Справочные материалы](#)

[SystemParametersInfo](#)

[TRACKMOUSEEVENT](#)

[_TrackMouseEvent](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Структура TRACKMOUSEEVENT (winuser.h)

Статья 28.08.2023

Используется функцией [TrackMouseEvent](#) для отслеживания того, когда указатель мыши покидает окно или наносит указатель мыши на окно в течение определенного периода времени.

Синтаксис

C++

```
typedef struct tagTRACKMOUSEEVENT {
    DWORD cbSize;
    DWORD dwFlags;
    HWND  hwndTrack;
    DWORD dwHoverTime;
} TRACKMOUSEEVENT, *LPTRACKMOUSEEVENT;
```

Члены

cbSize

Тип: **DWORD**

Размер структуры **TRACKMOUSEEVENT** в байтах.

dwFlags

Тип: **DWORD**

Запрошенные службы. Этот элемент может быть сочетанием следующих значений.

Значение	Значение
TME_CANCEL 0x80000000	Вызывающий объект хочет отменить предыдущий запрос на отслеживание. Вызывающий объект также должен указать тип отслеживания, которое требуется отменить. Например, чтобы отменить отслеживание при наведении, вызывающий объект должен передать флаги TME_CANCEL и TME_HOVER .

TME_HOVER 0x00000001	Вызывающему объекту требуется уведомление о наведении указателя мыши. Уведомление доставляется в виде сообщения WM_MOUSEHOVER . Если вызывающий объект запрашивает отслеживание при наведении, а отслеживание наведении на него уже активно, таймер наведении на него будет сброшен.
	Этот флаг игнорируется, если указатель мыши не находится на указанном окне или области.
TME_LEAVE 0x00000002	Вызывающий объект хочет оставить уведомление. Уведомление доставляется в виде сообщения WM_MOUSELEAVE . Если указатель мыши не находится над указанным окном или областью, уведомление о выходе создается немедленно и дальнейшее отслеживание не выполняется.
TME_NONCLIENT 0x00000010	Вызывающий объект хочет наведите указатель мыши и оставить уведомление для неклиентных областей. Уведомления доставляются в виде WM_NCMOUSEHOVER и WM_NCMOUSELEAVE сообщений.
TME_QUERY 0x40000000	Функция заполняет структуру вместо того, чтобы рассматривать ее как запрос отслеживания. Структура заполняется таким образом, что если бы эта структура была передана в TrackMouseEvent , она бы создавала текущее отслеживание. Единственная аномалия заключается в том, что возвращаемое время ожидания при наведении курсора всегда является фактическим временем ожидания, а не HOVER_DEFAULT , если HOVER_DEFAULT было указано во время исходного запроса TrackMouseEvent .

`hwndTrack`

Тип: **HWND**

Дескриптор для отслеживаемого окна.

`dwHoverTime`

Тип: **DWORD**

Время ожидания при наведении (если **TME_HOVER** было указано в **dwFlags**) в миллисекундах. Может быть **HOVER_DEFAULT**, что означает использование времени ожидания наведении указателя мыши по умолчанию.

Комментарии

Время ожидания наведении по умолчанию по умолчанию — это время раскрывающегося меню, которое составляет 400 миллисекунда. Вы можете вызвать [SystemParametersInfo](#) и использовать **SPI_GETMOUSEHOVERTIME** для получения времени ожидания наведения по умолчанию.

Системный прямоугольник наведение по умолчанию совпадает с прямоугольником двойного щелчка. Вы можете вызвать [SystemParametersInfo](#) и использовать **SPI_GETMOUSEHOVERWIDTH** и **SPI_GETMOUSEHOVERHEIGHT**, чтобы получить размер прямоугольника, в котором должен оставаться указатель мыши для [TrackMouseEvent](#) для создания [сообщения WM_MOUSEOVER](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Верхняя часть	winuser.h (включая Windows.h)

См. также раздел

[Ввод с помощью мыши](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция UnloadKeyboardLayout (winuser.h)

Статья 28.08.2023

Выгрузка идентификатора входного языкового стандарта (прежнее название — раскладка клавиатуры).

Синтаксис

C++

```
BOOL UnloadKeyboardLayout(  
    [in] HKL hkl  
) ;
```

Параметры

[in] hkl

Тип: HKL

Идентификатор входного языкового стандарта для выгрузки.

Возвращаемое значение

Тип: BOOL

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение. Функция может завершиться ошибкой по следующим причинам:

- Был передан недопустимый идентификатор входного языкового стандарта.
- Идентификатор входного языкового стандарта был предварительно загружен.
- Используется идентификатор входного языкового стандарта.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

UnloadKeyboardLayout не может выгрузить системный идентификатор входного языкового стандарта по умолчанию, если он является единственной загруженной раскладкой клавиатуры. Прежде чем выгрузить входной языковой стандарт, необходимо сначала загрузить другой идентификатор входного языкового стандарта.

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[ActivateKeyboardLayout](#)

Основные понятия

[GetKeyboardLayoutName](#)

[Ввод с клавиатуры](#)

[LoadKeyboardLayout](#)

Справочные материалы

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция UnregisterHotKey (winuser.h)

Статья 28.08.2023

Освобождает горячий ключ, ранее зарегистрированный вызывающим потоком.

Синтаксис

C++

```
BOOL UnregisterHotKey(  
    [in, optional] HWND hWnd,  
    [in]           int   id  
)
```

Параметры

[in, optional] hWnd

Тип: **HWND**

Дескриптор окна, связанного с горячей клавишей для освобождения. Этот параметр должен иметь значение **NULL**, если горячая клавиша не связана с окном.

[in] id

Тип: **int**

Идентификатор освобождаемого горячего ключа.

Возвращаемое значение

Тип: **BOOL**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Дополнительные сведения об ошибке можно получить, вызвав [GetLastError](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

[Основные понятия](#)

[Ввод с клавиатуры](#)

[Справочные материалы](#)

[RegisterHotKey](#)

[WM_HOTKEY](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция VkKeyScanA (winuser.h)

Статья 28.08.2023

[Эта функция заменена функцией [VkKeyScanEx](#). Однако вы по-прежнему можете использовать [VkKeyScan](#), если не нужно указывать раскладку клавиатуры.]

Преобразует символ в соответствующий код виртуальной клавиши и состояние сдвига для текущей клавиатуры.

Синтаксис

C++

```
SHORT VkKeyScanA(  
    [in] CHAR ch  
)
```

Параметры

[in] ch

Тип: **TCHAR**

Символ, который необходимо преобразовать в код виртуального ключа.

Возвращаемое значение

Тип: **SHORT**

Если функция выполняется успешно, байт возвращаемого значения в нижнем порядке содержит код виртуального ключа, а байт высокого порядка — состояние сдвига, которое может быть сочетанием следующих битов флага.

Возвращаемое значение	Описание
1	Нажата любая клавиша SHIFT.
2	Нажата либо клавиша CTRL.
4	Нажата клавиша ALT.
8	Нажата клавиша Ханкаку

16	Зарезервировано (определяется драйвером раскладки клавиатуры).
32	Зарезервировано (определяется драйвером раскладки клавиатуры).

Если функция не находит ключ, который преобразуется в код переданного символа, байты нижнего и высокого порядка содержат –1.

Комментарии

Для раскладок клавиатуры, где в качестве клавиши SHIFT используется правая клавиша ALT (например, французская раскладка клавиатуры), состояние shift представлено значением 6, так как правая клавиша ALT преобразуется внутри в ctrl+ALT.

Переводы цифровой клавиатуры (`VK_NUMPAD0` через `VK_DIVIDE`) игнорируются. Эта функция предназначена для преобразования символов в нажатия клавиш только из раздела main клавиатуры. Например, символ "7" преобразуется в `VK_7`, а не в `VK_NUMPAD7`.

`VkKeyScan` используется приложениями, которые отправляют символы с помощью `WM_KEYUP` и `WM_KEYDOWN` сообщений.

ⓘ Примечание

Заголовок `winuser.h` определяет `VkKeyScan` в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Использование псевдонима, не зависящий от кодирования, с кодом, который не является нейтральным для кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или времени выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия
клиента

Windows 2000 Professional [только классические
приложения]

Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyNameText](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [Ввод с клавиатуры](#)
- [SetKeyboardState](#)
- [VkKeyScanEx](#)
- [WM_KEYDOWN](#)
- [WM_KEYUP](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция VkKeyScanExA (winuser.h)

Статья 28.08.2023

Преобразует символ в соответствующий код виртуального ключа и состояние сдвига. Функция переводит символ с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода.

Синтаксис

C++

```
SHORT VkKeyScanExA(  
    [in] CHAR ch,  
    [in] HKL dwhkl  
)
```

Параметры

[in] ch

Тип: TCHAR

Символ, который необходимо преобразовать в код виртуального ключа.

[in] dwhkl

Тип: HKL

Идентификатор языкового стандарта, используемый для перевода символа. Этот параметр может быть любым идентификатором входного языкового стандарта, ранее возвращенным функцией [LoadKeyboardLayout](#).

Возвращаемое значение

Тип: SHORT

Если функция выполняется успешно, байт возвращаемого значения в нижнем порядке содержит код виртуального ключа, а байт высокого порядка — состояние сдвига, которое может быть сочетанием следующих битов флага.

Возвращаемое значение	Описание
1	Нажата любая клавиша SHIFT.
2	Нажата либо клавиша CTRL.
4	Нажата клавиша ALT.
8	Нажата клавиша Ханкаку
16	Зарезервировано (определяется драйвером раскладки клавиатуры).
32	Зарезервировано (определяется драйвером раскладки клавиатуры).

Если функция не находит ключ, который преобразуется в код переданного символа, байты нижнего и высокого порядка содержат –1.

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Для раскладок клавиатуры, где в качестве клавиши SHIFT используется правая клавиша ALT (например, французская раскладка клавиатуры), состояние shift представлено значением 6, так как правая клавиша ALT преобразуется внутри в ctrl+ALT.

Переводы цифровой клавиатуры (VK_NUMPAD0 через VK_DIVIDE) игнорируются. Эта функция предназначена для преобразования символов в нажатия клавиш только из раздела main клавиатуры. Например, символ "7" преобразуется в VK_7, а не в VK_NUMPAD7.

VkKeyScanEx используется приложениями, которые отправляют символы с помощью сообщений WM_KEYUP и WM_KEYDOWN .

ⓘ Примечание

Заголовок winuser.h определяет VkKeyScanEx в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Использование псевдонима, не зависящий от кодирования, с кодом, который не является

нейтральным для кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или времени выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyNameText](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [LoadKeyboardLayout](#)
- [SetKeyboardState](#)
- [ToAsciiEx](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция VkKeyScanExW (winuser.h)

Статья 28.08.2023

Преобразует символ в соответствующий код виртуального ключа и состояние сдвига. Функция преобразует символ с помощью языка ввода и физической раскладки клавиатуры, определяемой идентификатором языкового стандарта ввода.

Синтаксис

C++

```
SHORT VkKeyScanExW(  
    [in] WCHAR ch,  
    [in] HKL     dwhkl  
) ;
```

Параметры

[in] ch

Тип: TCHAR

Символ, который необходимо преобразовать в код виртуального ключа.

[in] dwhkl

Тип: HKL

Идентификатор входного языкового стандарта, используемый для перевода символа. Этот параметр может быть любым идентификатором входного языкового стандарта, ранее возвращенным функцией [LoadKeyboardLayout](#).

Возвращаемое значение

Тип: SHORT

Если функция выполнена успешно, байт низкого порядка возвращаемого значения содержит код виртуального ключа, а байт высокого порядка — состояние сдвига, которое может быть сочетанием следующих битов флага.

Возвращаемое значение	Описание
1	Нажата любая клавиша SHIFT.
2	Нажата либо клавиша CTRL.
4	Нажата либо клавиша ALT.
8	Нажата клавиша Ханкаку
16	Зарезервировано (определяется драйвером раскладки клавиатуры).
32	Зарезервировано (определяется драйвером раскладки клавиатуры).

Если функция не находит ключ, который преобразуется в переданный код символов, байты низкого и высокого порядка содержат –1.

Комментарии

Идентификатор языкового стандарта ввода является более широким понятием, чем раскладка клавиатуры, так как он также может охватывать преобразователь речи в текст, редактор метода ввода (IME) или любую другую форму ввода.

Для раскладок клавиатуры, где в качестве клавиши shift используется правая клавиша ALT (например, французская раскладка клавиатуры), состояние shift представлено значением 6, так как правая клавиша ALT преобразуется внутри в сочетание клавиш CTRL+ALT.

Переводы цифровой клавиатуры (VK_NUMPAD0 по VK_DIVIDE) игнорируются. Эта функция предназначена для преобразования символов в нажатия клавиш только из раздела main клавиатуры. Например, символ "7" преобразуется в VK_7, а не в VK_NUMPAD7.

`VkKeyScanEx` используется приложениями, которые отправляют символы с помощью [WM_KEYUP](#) и [WM_KEYDOWN](#) сообщений.

ⓘ Примечание

Заголовок `winuser.h` определяет `VkKeyScanEx` как псевдоним, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Сочетание использования псевдонима, не зависящий от кодировки, с кодом, не зависящим от

кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или среды выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия клиента	Windows 2000 Professional [только классические приложения]
Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyNameText](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [LoadKeyboardLayout](#)
- [SetKeyboardState](#)
- [ToAsciiEx](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)

Функция VkKeyScanW (winuser.h)

Статья 28.08.2023

[Эта функция заменена функцией [VkKeyScanEx](#). Однако вы по-прежнему можете использовать [VkKeyScan](#), если не нужно указывать раскладку клавиатуры.]

Преобразует символ в соответствующий код виртуальной клавиши и состояние сдвига для текущей клавиатуры.

Синтаксис

C++

```
SHORT VkKeyScanW(  
    [in] WCHAR ch  
)
```

Параметры

[in] ch

Тип: **TCHAR**

Символ, который необходимо преобразовать в код виртуального ключа.

Возвращаемое значение

Тип: **SHORT**

Если функция выполняется успешно, байт возвращаемого значения в нижнем порядке содержит код виртуального ключа, а байт высокого порядка — состояние сдвига, которое может быть сочетанием следующих битов флага.

Возвращаемое значение	Описание
1	Нажата любая клавиша SHIFT.
2	Нажата либо клавиша CTRL.
4	Нажата клавиша ALT.
8	Нажата клавиша Ханкаку

16	Зарезервировано (определяется драйвером раскладки клавиатуры).
32	Зарезервировано (определяется драйвером раскладки клавиатуры).

Если функция не находит ключ, который преобразуется в код переданного символа, байты нижнего и высокого порядка содержат –1.

Комментарии

Для раскладок клавиатуры, где в качестве клавиши SHIFT используется правая клавиша ALT (например, французская раскладка клавиатуры), состояние shift представлено значением 6, так как правая клавиша ALT преобразуется внутри в ctrl+ALT.

Переводы цифровой клавиатуры (`VK_NUMPAD0` через `VK_DIVIDE`) игнорируются. Эта функция предназначена для преобразования символов в нажатия клавиш только из раздела main клавиатуры. Например, символ "7" преобразуется в `VK_7`, а не в `VK_NUMPAD7`.

`VkKeyScan` используется приложениями, которые отправляют символы с помощью `WM_KEYUP` и `WM_KEYDOWN` сообщений.

ⓘ Примечание

Заголовок `winuser.h` определяет `VkKeyScan` в качестве псевдонима, который автоматически выбирает версию ANSI или Юникод этой функции на основе определения константы препроцессора ЮНИКОД. Использование псевдонима, не зависящий от кодирования, с кодом, который не является нейтральным для кодировки, может привести к несоответствиям, которые приводят к ошибкам компиляции или времени выполнения. Дополнительные сведения см. в разделе [Соглашения для прототипов функций](#).

Требования

Минимальная версия
клиента

Windows 2000 Professional [только классические
приложения]

Минимальная версия сервера	Windows 2000 Server [только классические приложения]
Целевая платформа	Windows
Header	winuser.h (включая Windows.h)
Библиотека	User32.lib
DLL	User32.dll

См. также раздел

- [GetAsyncKeyState](#)
- [GetKeyNameText](#)
- [GetKeyState](#)
- [GetKeyboardState](#)
- [SetKeyboardState](#)
- [VkKeyScanEx](#)
- [WM_KEYDOWN](#)
- [WM_KEYUP](#)
- [Ввод с клавиатуры](#)

Обратная связь

Были ли сведения на этой странице полезными?

 Да

 Нет

[Получить справку в Microsoft Q&A](#)