



119270, Москва, Лужнецкая наб., д. 6,
стр.1, офис 214, ООО «ЭР СИ О»
Тел./факс: (495) 287-98-87
E-mail: info@rco.ru
<http://www.rco.ru>

**Руководство разработчика
RCO Morphology – библиотека
морфологического анализа**

Версия 2.1

(Microsoft Windows, Linux i386)

Москва, 2013

В содержание данного документа могут быть внесены изменения без предварительного уведомления. Названия организаций, имена и даты, используемые в качестве примеров, являются вымышленными, если не оговорено обратное.

© ООО «ЭР СИ О», 2007-2013. Все права защищены.

ЭР СИ О, Russian Context Optimizer, RCO являются охраняемыми товарными знаками.

Oracle 9i, Oracle 10g, Oracle 11g, Oracle, Oracle Text, SQL*Plus, PL/SQL являются зарегистрированными торговыми марками корпорации Oracle.

ООО «ЭР СИ О» может являться правообладателем патентов и заявок, поданных на получение патента, товарных знаков и объектов авторского права, которые имеют отношение к содержанию данного документа.

Предоставление вам данного документа не означает передачи какой-либо лицензии на использование данных патентов, товарных знаков и объектов авторского права, за исключением использования, явно оговоренного в лицензионном соглашении ООО «ЭР СИ О».

Все другие названия юридических лиц и изделий являются охраняемыми товарными знаками или товарными знаками, принадлежащими их владельцам.

Содержание

Обзор	4
Отличия от предыдущих версий	5
Отличия от версии 2.0	5
Отличия от версии 1.0	5
Принципы работы	6
Система описания словоизменения	6
Точный морфологический анализ	6
Морфологический анализ на основе правил	7
Приближенный морфологический анализ	7
Интерфейс библиотеки	8
Загрузка/выгрузка словаря	9
MoInitialize	9
MoUninitialize	10
Функции для использования в ИПС	11
MoGetNormalForm	11
MoGetAllNormalForms	12
MoGetStem	14
MoGetAllForms	16
Функции для использования в приложениях анализа текста	19
MoAllocInfo	20
MoFreeInfo	20
MoAnalyze	20
MoQueryInfo	22
Работа с MoQueryInfo	24
Тип проведенного анализа	25
Число вариантов разбора	25
Нормальная форма слова	26
Основа слова	26
Длина распознанного конца слова	27
Код части речи	27
Название части речи	28
Признак одушевленности	28
Число вариантов грамматических форм слова	29
Код парадигмы словоизменения	29
Индекс грамматической формы в парадигме словоизменения	30
Максимально возможное число словоформ	30
Название грамматической формы	31
Заданная форма слова	31
Пример использования	33
Исходный код тестового примера	33
Результат работы тестового примера	37
Приложение 1. Описание парадигмы словоизменения	40
Приложение 2. Классификация частей речи	45

Обзор

Библиотека полного морфологического анализа слов русского языка позволяет решать следующие задачи:

- определять все грамматические характеристики словоформ (часть речи, падеж, спряжение и т.п.) и лексико-семантические разряды (имя, отчество, фамилия, наименование организации, географическое название);
- приводить различные грамматические формы слова к нормальной (словарной) форме;
- получать все грамматические формы слова;
- проверять орфографию.

Библиотека включает в себя:

- точный анализ известных слов по словарю объемом более 115 тысяч слов, что покрывает более 3-х миллионов словоформ;
- высоко достоверный анализ неизвестного слова на основе комплекса правил словообразования и словоизменения;
- вероятностный анализ посредством соотнесения с моделями словоизменения часто встречающихся слов на основе оценки флективной и суффиксальной частей слова.

Подход к описанию морфологической системы языка использует обучение на примерах словоизменения, в ходе которого автоматически выделяются окончания, основы, суффиксы, и строятся модели, описывающие изменение различных классов слов.

Объем бинарного словаря – 3 Мб.

Скорость морфологического анализа – более 100 тысяч слов в секунду (процессор **AMD Athlon**, 1000 МГц).

Библиотека поставляется в вариантах: **Windows** (win32) и **Linux** (i386).

Обрабатываемая кодировка текста – **Win1251**. Библиотека также поддерживает работу с **UNICODE** (только для **Windows**).

В состав библиотеки входят следующие файлы:

gpmorph.h	файл заголовка с описанием констант и прототипов функций
gpmorph.lib	lib-файл для компоновки в среде MS Visual C++ 6.0 (только в Windows -версии)
gpmorph.dll	динамическая библиотека (только в Windows -версии)
libgpmorph.so	динамическая библиотека (только в Linux -версии)
morphdct.dat	бинарный морфологический словарь
mosample.cpp	исходный код тестового примера на C++

Отличия от предыдущих версий

Ниже перечислены отличия данной версии библиотеки от ее предыдущих версий.

Отличия от версии 2.0

Незначительные изменения в интерфейсе библиотеки упрощают ее использование в информационно-поисковых системах, позволяя легко выделять наиболее вероятный вариант разбора слова в случае омонимии:

- введены новые параметры вызова функции **MoGetNormalForm**, которая предназначена для построения единственной нормальной формы слова;
- добавлен новый параметр вызова функции **MoGetAllForms** (*MO_O_EXPANDBESTNORMALFORM*), которая предназначена для построения всех грамматических форм слова.

Отличия от версии 1.0

- добавлен [Морфологический анализ на основе правил](#);
- в ряд функций добавлена группа параметров, позволяющих учитывать при анализе особенности написания слова;

MO_O_LOWER_NONAME_NAME

Слово, написанное с маленькой буквы, всегда может быть опознано как имя собственное.

MO_O_LOWER_NONAME_NAME_NOHOMONIMS

Слово, написанное с маленькой буквы, может быть опознано как имя собственное, только если оно не опознано как имя нарицательное.

MO_O_LOWER_NONAME

Слово, написанное с маленькой буквы, не может быть опознано как имя собственное.

MO_O_UPPER_NAME_NONAME

Слово, написанное с большой буквы, может быть опознано как имя собственное, и как нарицательное.

MO_O_UPPER_NAME

Слово, написанное с большой буквы, не может быть опознано только как имя собственное.

- в ряд функций добавлена группа параметров, позволяющих влиять на обработку неизвестных слов.

MO_O_NORMALFORM

Предполагать, что слово стоит в нормальной форме.

MO_O_NAME

Предполагать, что слово является именем собственным.

MO_O_NONAME

Предполагать, что слово является именем нарицательным.

MO_O_PLURAL_IMPOSSIBLE

Предполагать, что слово стоит в форме единственного числа.

Принципы работы

Система описания словоизменения

При создании библиотеки морфологического анализа русского языка была принята универсальная система описания словоизменения, в максимальной степени независимая от языка.

Слово представляется графической основой, наиболее длинной цепочкой символов, общей у всех форм слова. При этом возможны нулевые основы (*идти – шел*).

Изменяемая часть слова описывается набором присоединяемых к основе окончаний. Список окончаний, упорядоченных в соответствии с представляемыми ими грамматическими формами, в принятой здесь терминологии будет называться парадигмой (или моделью) словоизменения.

В русском языке существует четыре типа парадигм (см. [Приложение 1](#)): существительного (14 грамматических форм), прилагательного (31 грамматическая форма) и глагола (146 форм, не считая 86 возвратных). К четвертому типу (вырожденному) относятся все неизменяемые слова.

Большинство слов языка изменяется стандартно: имеет одинаковые окончания в одинаковых грамматических формах. Поэтому все парадигмы собраны отдельно от основ в трех таблицах парадигм (для существительных, прилагательных, глаголов). При основе слова хранится ссылка на нужную парадигму в соответствующей таблице, что позволяет восстановить любую из словоформ, приписывая подходящее окончание к основе.

Окончания из парадигм обычно также являются стандартными, поэтому они хранятся в общей таблице окончаний, а в парадигмах – ссылки на соответствующие окончания.

Таким образом, каждое слово словаря описывается основой, типом парадигмы словоизменения и номером парадигмы в соответствующей таблице парадигм. Такая система обеспечивает компактное описание словаря, в результате которого 30 Мб, соответствующие трем миллионам словоформ, пакуются в 3 Мб.

Для построения заданной грамматической формы по известным основе и парадигме достаточно выбрать номер соответствующего окончания из парадигмы и получить его строку из таблицы окончаний, после чего приписать к основе.

Точный морфологический анализ

Для быстрого поиска словоформы в словаре морфоанализа создается дополнительная структура – дерево окончаний. Дерево окончаний дублирует все окончания, представленные в таблице окончаний в форме строк. Объем дерева окончаний по сравнению с объемом дерева основ незначителен.

Поиск словоформы реализуется следующим образом:

- Слово анализируется с конца на совпадение с деревом окончаний;
- Для каждого совпавшего окончания, включая пустое, оставшаяся часть слова анализируется в дереве основ;
- При наличии полного совпадения остатка слова с некоторой основой сравниваются списки парадигм при окончании и основе. Если в списках обнаруживается общий элемент, значит, окончание возможно при данной основе, и словоформа распознается.

Морфологический анализ на основе правил

Если слово не распознано как известное в морфологическом словаре, автоматически делается попытка проанализировать его префикс и постфикс.

При этом, во-первых, распознаются слова, образованные от известных за счет прибавления известных префиксов, например, приставка «*трех*» или «*трех-*» может предшествовать прилагательному, а префикс «*авиа*» – существительному или прилагательному.

Во-вторых, анализируется постфикс слова, который в общем случае включает неизменяемый суффикс и изменяемое окончание. Например, наличие суффиксов «*штейн*» или «*енко*» вместе с соответствующими окончаниями при слове, написанном с заглавной буквы, позволяет однозначно идентифицировать его как фамилию, наличие суффикса «*горск*» – как географическое наименование мужского рода, а «*ович*» и «*евич*» – как фамилию или отчество.

В состав словаря словообразования входят несколько сотен часто употребляемых приставок для различных частей речи и несколько сотен суффиксов с соответствующими парадигмами словоизменения, которые позволяют распознать неизвестное слово и определить все его характеристики с очень высокой степенью достоверности.

Учитываются и другие, более сложные правила словообразования.

Приближенный морфологический анализ

Алгоритм приближенного морфоанализа отличается от точного анализа тем, что вместо дерева основ используется дерево суффиксов, автоматически сформированное на этапе компиляции словаря. При этом для каждого совпавшего окончания находится самое длинное совпадение конца оставшейся части слова с одним из суффиксов (при условии наличия при суффиксе и окончании одинаковой парадигмы словоизменения).

В качестве наиболее вероятной парадигмы изменения принимается та, при которой суммарная длина суффикса и окончания оказывается наибольшей. При наличии нескольких кандидатов равной длины приоритет имеет парадигма с большей частотой встречаемости.

Для повышения точности приближенного морфоанализа в расчет принимаются парадигмы, имеющие частоту встречаемости не менее некоторого порога, определенного эмпирически.

Результатом приближенного морфоанализа является основа слова (длиной не менее 2-х символов) и наиболее вероятная парадигма изменения, позволяющая построить все словоформы.

Интерфейс библиотеки

Описание интерфейса библиотеки сгруппировано по разделам. В первом разделе приведены функции для работы со словарем. Они необходимы при любом варианте использования библиотеки. Во втором разделе приведен минимальный набор базовых функций, предназначенный для встраивания в информационно-поисковые системы (ИПС). Функции этого раздела не позволяют получить грамматическую информацию о слове, но оптимизированы для типичных задач, возникающих при построении ИПС. В двух последних разделах приводится описание функций, позволяющих получить всю грамматическую информацию о слове. Эти функции предназначены преимущественно для использования в лингвистическом ПО и специальных системах анализа текста, доступны только в версии библиотеки **RCO Morphology Professional**.

[Загрузка/выгрузка словаря](#)

[Функции для использования в ИПС](#)

[Функции для использования в приложениях анализа текста](#)

[Работа с **MoQueryInfo**](#)

Загрузка/выгрузка словаря

Перед началом работы словарь морфологического анализа должен быть загружен, а по окончании – выгружен. Следующие функции обеспечивают загрузку и выгрузку словаря.

- [MoInitialize](#)
- [MoUninitialize](#)

MoInitialize

Данная функция используется для загрузки словаря и должна быть вызвана перед началом работы.

```
int MoInitialize(  
TCHAR* pszDict // путь к файлу со словарем  
);
```

Параметры

pszDict

[вх] Путь к файлу со словарем. Если указаны путь и имя файла, производится загрузка по указанному пути. Если задан каталог, а имя не задано, загружается файл **morphdct.dat**, который должен находиться в заданном каталоге.

Если параметр *pszDict* равен NULL, либо указывает на пустую строку ($0 == *pszDict$), либо содержит только имя файла, библиотека пытается найти файл (**morphdct.dat**, если не задано иное), перебирая каталоги в следующем порядке.

1. Каталог, указанный в переменной окружения **GPSN_PATH** (только в **Linux**-версии);
2. Каталог с библиотекой **gpmorph.dll** (только в **Windows**-версии);
3. Каталог с исполняемым файлом, который владеет текущим процессом (только в **Windows**-версии);
4. Текущий каталог;
5. Системный каталог **Windows (system32)** (только в **Windows**-версии);
6. Каталог **Windows** (только в **Windows**-версии);
7. Каталоги из переменной окружения **PATH**.

При этом для каждого найденного файла делается попытка открыть файл со словарем, пока загрузка не завершится успешно.

Возвращаемое значение

В случае успешной загрузки функция возвращает *0*, иначе – *1*. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Linux**-версии в качестве кода ошибки используйте значение переменной **errno**.

Замечание. Загрузка словаря осуществляется только при первом обращении к функции. Последующие вызовы увеличивают счетчик клиентов библиотеки, который уменьшается при вызове [MoUninitialize](#).

См. также: [MoUninitialize](#).

MoUninitialize

Данная функция задействуется для выгрузки словаря из памяти и освобождения ресурсов, используемых библиотекой. Должна вызываться по окончании работы.

```
int MoUninitialize(  
);
```

Возвращаемое значение

В случае успешной выгрузки функция возвращает *0*, иначе – *1*. В случае неудачи вызывайте `GetLastError` для получения кода ошибки. В **Linux**-версии в качестве кода ошибки используйте значение переменной `errno`.

Замечание. Выгрузка словаря осуществляется только при обращении к функции последнего клиента библиотеки. Предыдущие вызовы уменьшают счетчик клиентов библиотеки, который увеличивается при вызове **MoInitialize**.

См. также: **MoInitialize**

Функции для использования в ИПС

Функции этого раздела предназначены для стандартного использования в информационно-поисковых системах в случаях, когда необходимо получение нормальной формы (или основы) слова или построение всех словоформ.

- [MoGetNormalForm](#)
- [MoGetAllNormalForms](#)
- [MoGetStem](#)
- [MoGetAllForms](#)

MoGetNormalForm

Данная функция используется для получения единственной нормальной формы слова по его произвольной форме. При этом делается попытка точного морфологического анализа с поиском в словаре известных слов, а затем, в случае неудачи, производится обработка слова по правилам.

```
int MoGetNormalForm(  
    TCHAR* pszWord,    // анализируемое слово  
    TCHAR* pszBuf,    // буфер с нормальной формой  
    int* pnBufSize,    // размер возвращаемого буфера  
    unsigned uOptions // параметры алгоритма анализа  
);
```

Параметры

pszWord

[вх] Слово для анализа. Указатель на буфер со строкой, завершающейся нулем.

pszBuf

[вых] Указатель на буфер для получения строки, заканчивающейся нулем и содержащей единственную нормальную форму слова. В случае морфологической омонимии, когда слову может быть сопоставлено несколько нормальных форм, возвращаемая нормальная форма определяется в соответствии с комплексом правил, которые [позволяют](#) отобразить наиболее вероятный вариант.

Если *pszBuf* равен *NULL*, а *pnBufSize* не равен *NULL*, функция завершается с кодом 0, и по адресу, на который указывает переменная *pnBufSize*, помещается число символов, включая завершающий ноль, необходимое для записи нормальной формы.

pnBufSize

[вх/вых] Указатель на переменную, что содержит размер буфера *pszBuf*, выражаемый в количестве символов. На выходе эта переменная содержит число байт, включая завершающий ноль, которое было скопировано в буфер.

Если размер буфера недостаточен, функция устанавливает код ошибки *ERROR_MORE_DATA* и записывает необходимый размер, выражаемый в количестве символов, по адресу, на который указывает *pnBufSize*. В этом случае содержимое буфера *pszBuf* не определено.

Во всех случаях возвращаемое в *pnBufSize* значение содержит размер, выражаемый в количестве символов, включая завершающий ноль.

uOptions

[вх] Параметры алгоритма. Допускаются два значения:

MO_O_OPTIMALQUERYPARSING Оптимальный режим для обработки слов из поискового запроса.

MO_O_OPTIMALTEXTPARSING Оптимальный режим для обработки слов из текста.

Возвращаемое значение

В случае успешного завершения функция возвращает общее число нормальных форм, найденных в словаре, иначе – 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Linux**-версии в качестве кода ошибки используйте значение переменной `errno`.

Замечания.

Функция работает со строками в кодировке **Win1251**, и в качестве типа *TCHAR* нужно применять *char*. Если объявлен макрос **UNICODE**, функция предполагает, что используются символы страницы 1049, и в качестве типа *TCHAR* нужно применять *short*.

Капитализация первой буквы влияет на анализ. Например, при сохранении исходных параметров нормальная форма для слова «*коля*» будет определена как «*КОЛОТЬ*», а для слова «*Коля*» – как «*КОЛЯ*» (либо как фамилия «*КОЛЬ*»). Слово же «*россия*» (имя собственное), написанное с маленькой буквы, распознается, поскольку имеет единственный вариант разбора. На выход возвращается слово из заглавных букв, в котором буква «*Ё*» заменена на «*Е*».

Остаток слова, превышающий константу *MO_MAXWORDLEN*, отсекается.

Рекомендуемый размер буфера *pszBuf* задается константой *MO_SMALLBUFSIZE*.

Если функция вернула число большее 1, полный список нормальных форм представит функция **MoGetAllNormalForms**.

Если функция вернула 0, это означает, что точный морфологический анализ невозможен – слово не найдено в словаре и его не удастся достоверно проанализировать по правилам. В этом случае в буфер *pszBuf* помещается исходное слово.

В качестве аналога нормальной формы для неизвестных слов рекомендуется попытаться получить основу слова при помощи **MoGetStem**.

См. также: **MoGetAllNormalForms**, **MoGetStem**, **MoGetAllForms**

MoGetAllNormalForms

Данная функция используется для получения всех нормальных форм слова (различных по начертанию) по его произвольной форме. При этом делается попытка точного морфологического анализа. Так, для слова «*чаще*» будет получено несколько нормальных форм: наречие «*чаще*», прилагательное «*частый*» и существительное «*чаща*».

```
int MoGetAllNormalForms (
    TCHAR* pszWord,    // анализируемое слово
    TCHAR* pszBuf,    // буфер с нормальными формами
    int* pnBufSize,   // размер возвращаемого буфера
    unsigned uOptions // параметры алгоритма анализа
);
```

Параметры

pszWord

[вх] Слово для анализа. Указатель на буфер со строкой, завершающейся нулем.

pszBuf

[вых] Указатель на буфер для получения строки, заканчивающейся нулем и содержащей все различные по начертанию нормальные формы слова. Нормальные формы разделяются нулем, а последовательность нормальных форм завершают два нуля.

Если *pszBuf* равен *NULL*, а *pnBufSize* не равен *NULL*, функция завершается с кодом 0, и по адресу, на который указывает переменная *pnBufSize*, помещается число символов, включая завершающий ноль, необходимое для записи нормальной формы.

pnBufSize

[вх/вых] Указатель на переменную, что содержит размер буфера *pszBuf*, выражаемый в количестве символов. На выходе эта переменная содержит число байт, включая завершающий ноль, которое было скопировано в буфер.

Если размер буфера недостаточен, функция устанавливает код ошибки *ERROR_MORE_DATA* и записывает необходимый размер, выражаемый в количестве символов, по адресу, на который указывает *pnBufSize*. В этом случае содержимое буфера *pszBuf* не определено.

Во всех случаях возвращаемое в *pnBufSize* значение содержит размер, выражаемый в количестве символов, включая завершающий ноль.

uOptions

[вх] Параметры алгоритма при анализе имен собственных. Допускается комбинация следующих битовых флагов:

<i>MO_O_LOWER_NONAME_NAME</i>	Слово, написанное с маленькой буквы, всегда может быть опознано как имя собственное.
<i>MO_O_LOWER_NONAME_NAME_NOHOMONIMS</i>	Слово, написанное с маленькой буквы, может быть опознано как имя собственное, только если оно не опознано как имя нарицательное.
<i>MO_O_LOWER_NONAME</i>	Слово, написанное с маленькой буквы, не может быть опознано как имя собственное.
<i>MO_O_UPPER_NAME_NONAME</i>	Слово, написанное с большой буквы, может быть опознано и как имя собственное, и как нарицательное.
<i>MO_O_UPPER_NAME</i>	Слово, написанное с большой буквы, не может быть опознано только как имя собственное.

Установка параметров *MO_O_LOWER_NONAME_NAME*, *MO_O_LOWER_NONAME_NAME_NOHOMONIMS*, *MO_O_LOWER_NONAME* является взаимоисключающей. То же относится к параметрам *MO_O_UPPER_NAME_NONAME* и *MO_O_UPPER_NAME*.

Если ни один из параметров не задан, то используется изначальная маска для значений, равная *MO_O_LOWER_NONAME_NAME_NOHOMONIMS* | *MO_O_UPPER_NAME_NONAME*.

Возвращаемое значение

В случае успешного завершения функция возвращает число нормальных форм, иначе – 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Linux**-версии в качестве кода ошибки используйте значение переменной *errno*.

Замечания.

Функция работает со строками в кодировке **Win1251**, и в качестве типа *TCHAR* нужно применять *char*. Если объявлен макрос **UNICODE**, функция предполагает, что используются символы страницы 1049, и в качестве типа *TCHAR* нужно применять *short*.

Капитализация первой буквы влияет на анализ. Например, при сохранении исходных параметров нормальная форма для слова «*коля*» будет определена как «*КОЛОТЬ*», а для слова «*Коля*» – как «*КОЛЯ*» (либо как фамилия «*КОЛЬ*»). Слово же «*россия*» (имя собственное), написанное с маленькой буквы, будет тем не менее распознано, так как имеет единственный вариант разбора. На выход возвращается слово из заглавных букв, в котором буква «*Ё*» заменена на «*Е*».

Остаток слова, превышающий константу *MO_MAXWORDLEN*, отсекается.

Рекомендуемый размер буфера *pszBuf* задается константой *MO_SMALLBUFSIZE*.

Если функция вернула 0, это означает, что точный морфологический анализ невозможен – слово не найдено в словаре и его не удастся достоверно проанализировать по правилам. В этом случае в буфер *pszBuf* помещается исходное слово.

В качестве аналога нормальной формы для неизвестных слов рекомендуется попытаться получить основу слова при помощи [MoGetStem](#).

См. также: [MoGetNormalForm](#), [MoGetStem](#), [MoGetAllForms](#)

MoGetStem

Данная функция служит для получения средствами приближенного морфологического анализа основы слова по его произвольной форме. Так, для слова «*глокая*» основой будет «*глок*». В силу особенностей языка точное определение нормальной формы произвольного неизвестного слова затруднено. Поэтому в качестве инварианта различных словоформ при информационном поиске целесообразно использовать именно словооснову.

```
int MoGetStem(
    TCHAR* pszWord,    // анализируемое слово
    TCHAR* pszBuf,    // буфер с основой слова
    int* pnBufSize,    // размер возвращаемого буфера
    unsigned uOptions // параметры алгоритма анализа
);
```

Параметры

pszWord

[вх] Слово для анализа. Указатель на буфер со строкой, завершающейся нулем.

pszBuf

[вых] Указатель на буфер для получения строки, заканчивающейся нулем и содержащей основу слова.

Если *pszBuf* равен *NULL*, а *pnBufSize* не равен *NULL*, функция завершается с кодом 0, и по адресу, на который указывает переменная *pnBufSize*, помещается число символов, включая завершающий ноль, необходимое для записи нормальной формы.

pnBufSize

[вх/вых] Указатель на переменную, что содержит размер буфера *pszBuf*, выражаемый в количестве символов. На выходе эта переменная содержит число символов, включая завершающий ноль, которое было скопировано в буфер.

Если размер буфера недостаточен, функция устанавливает код ошибки *ERROR_MORE_DATA* и записывает необходимый размер, выражаемый в количестве символов, по адресу, на который указывает *pnBufSize*. В этом случае содержимое буфера *pszBuf* не определено.

Во всех случаях возвращаемое в *pnBufSize* значение содержит размер, выражаемый в количестве символов, включая завершающий ноль.

uOptions

[вх] Параметры алгоритма при анализе неизвестных слов. Допускается использование комбинации следующих битовых флагов:

<i>MO_O_NORMALFORM</i>	Предполагать, что слово стоит в нормальной форме.
<i>MO_O_NAME</i>	Предполагать, что слово является именем собственным.
<i>MO_O_NONAME</i>	Предполагать, что слово является именем нарицательным.
<i>MO_O_PLURAL_IMPOSSIBLE</i>	Предполагать, что слово стоит в форме единственного числа.
<i>MO_O_SUBJECT</i>	Искать в парадигме существительных.
<i>MO_O_DEFINITION</i>	Искать в парадигме прилагательных.
<i>MO_O_VERB</i>	Искать в парадигме глаголов.
<i>MO_O_UNCHANGED</i>	Искать в парадигме неизменяемых слов.

Если ни один из параметров не задан, то используется исходная маска для значений, равная *MO_O_NAME* | *MO_O_NONAME* | *MO_O_SUBJECT* | *MO_O_DEFINITION* | *MO_O_VERB* | *MO_O_UNCHANGED*.

Возвращаемое значение

В случае успешного завершения функция возвращает число больше либо равное 0, иначе – 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Linux**-версии в качестве кода ошибки используйте значение переменной `errno`.

Замечания.

Функция работает со строками в кодировке Win1251, и в качестве типа *TCHAR* нужно применять *char*. Если объявлен макрос **UNICODE**, функция предполагает, что используются символы страницы 1049, и в качестве типа *TCHAR* нужно применять *short*.

Капитализация первой буквы не влияет на анализ. На выход поступает слово, состоящее целиком из заглавных букв, в котором буква «Ё» заменена на «Е».

Если длина слова превышает *MO_MAXWORDLEN*, остаток слова, превышающий данную константу, отсекается.

Рекомендуемый размер буфера *pszBuf*, выражаемый в количестве символов, задается константой *MO_SMALLBUFSIZE*.

Если основа получена в результате приближенного анализа, функция возвращает значение, большее или равное 1, которое равно длине опознанного постфикса, на основании которого была построена модель словоизменения данного слова. Чем больше это значение, тем более достоверен результат анализа.

Если приближенный анализ не удался, функция возвращает 0, а в буфер *pszBuf* помещается входное слово.

См. также: [MoGetNormalForm](#), [MoGetAllNormalForms](#), [MoGetAllForms](#)

MoGetAllForms

Данная функция предназначена для получения всех форм по произвольной форме слова. Сначала предпринимается попытка точного морфологического анализа, в случае неудачи – анализа по правилам, при повторной неудаче – приближенного анализа.

```
int MoGetAllForms(
    TCHAR* pszWord,    // анализируемое слово
    TCHAR* pszBuf,     // буфер с формами слова
    Int* pnBufSize,    // размер возвращаемого буфера
    unsigned uOptions // параметры алгоритма анализа
);
```

Параметры

pszWord

[вх] Слово для анализа. Указатель на буфер со строкой, завершающейся нулем.

pszBuf

[вых] Указатель на буфер для получения строки, заканчивающейся нулем и содержащей все различные по начертанию формы слова. Формы разделяются нулем, а последовательность нормальных форм завершают два последовательных нуля.

Если *pszBuf* равен *NULL*, а *pnBufSize* не равен *NULL*, функция завершается с кодом 0, и по адресу, на который указывает переменная *pnBufSize*, помещается число символов, включая завершающий ноль, необходимое для записи множества форм слова.

pnBufSize

[вх/вых] Указатель на переменную, что содержит размер буфера *pszBuf*, выражаемый в количестве символов. На выходе эта переменная содержит число символов, включая завершающий ноль, которое было скопировано в буфер.

Если размер буфера недостаточен, функция устанавливает код ошибки *ERROR_MORE_DATA* и записывает необходимый размер, выражаемый в количестве символов, по адресу, на который указывает *pnBufSize*. В этом случае содержимое буфера *pszBuf* не определено.

Во всех случаях возвращаемое в *pnBufSize* значение содержит размер, выражаемый в количестве символов, включая завершающий ноль.

uOptions

[вх] Параметры алгоритма при анализе и синтезе. Комбинация следующих битовых флагов влияет на точный анализ и синтез:

<i>MO_O_LOWER_NONAME_NAME</i>	Слово, написанное с маленькой буквы, всегда может быть опознано как имя собственное.
<i>MO_O_LOWER_NONAME_NAME_NOHOMONIMS</i>	Слово, написанное с маленькой буквы, может быть опознано как имя собственное только, если оно не опознано как имя нарицательное.
<i>MO_O_LOWER_NONAME</i>	Слово, написанное с маленькой буквы, не может быть опознано как имя собственное.
<i>MO_O_UPPER_NAME_NONAME</i>	Слово, написанное с большой буквы, может быть опознано и как имя собственное, и как нарицательное.
<i>MO_O_UPPER_NAME</i>	Слово, написанное с большой буквы, не может быть опознано только как имя собственное.

MO_O_EXPANDNAMEONLYIFFOUND Если среди вариантов разбора присутствует имя собственное, то синтез словоформ будет проведен только для данного варианта разбора.

MO_O_EXPANDBESTNORMALFORM В случае морфологической омонимии, когда существует несколько вариантов разбора слова, синтез словоформ будет производиться для единственного варианта, который считается наиболее вероятным. При установке этого флага все прочие флаги игнорируются

Установка флага *MO_O_EXPANDNAMEONLYIFFOUND* приводит к дополнительной фильтрации словоформ после применения всех указанных выше флагов.

Флаг *MO_O_EXPANDBESTNORMALFORM* перекрывает установку всех прочих флагов и обеспечивает выбор единственного наиболее вероятного варианта разбора слова при наличии омонимии или же в случае, когда слово неизвестно. Если данный флаг не установлен, то синтез словоформ производится для всех вариантов разбора, полученных в соответствии с прочими указанными выше флагами.

Установка параметров *MO_O_LOWER_NONAME_NAME*, *MO_O_LOWER_NONAME_NAME_NOHOMONIMS*, *MO_O_LOWER_NONAME* является взаимоисключающей. То же относится к параметрам *MO_O_UPPER_NAME_NONAME* и *MO_O_UPPER_NAME*.

Если ни один из параметров не задан, то используется изначальная маска, равная *MO_O_LOWER_NONAME_NAME_NOHOMONIMS | MO_O_UPPER_NAME_NONAME*.

Кроме того, допускается использование комбинации следующих битовых флагов, задающих параметры приближенного анализа в случае, если не задан флаг *MO_O_EXPANDBESTNORMALFORM*:

<i>MO_O_NORMALFORM</i>	Предполагать, что слово стоит в нормальной форме.
<i>MO_O_NAME</i>	Предполагать, что слово является именем собственным.
<i>MO_O_NONAME</i>	Предполагать, что слово является именем нарицательным.
<i>MO_O_SUBJECT</i>	Искать в парадигме существительных.
<i>MO_O_DEFINITION</i>	Искать в парадигме прилагательных.
<i>MO_O_VERB</i>	Искать в парадигме глаголов.
<i>MO_O_UNCHANGED</i>	Искать в парадигме неизменяемых слов.

Если ни один из параметров не задан, то используется изначальная маска для значений, равная *MO_O_NAME | MO_O_NONAME | MO_O_SUBJECT | MO_O_DEFINITION | MO_O_VERB | MO_O_UNCHANGED*.

Возвращаемое значение

В случае успешного завершения функция возвращает число построенных форм, иначе – 1. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В *Linux*-версии в качестве кода ошибки используйте значение переменной `errno`.

Замечания.

Функция работает со строками в кодировке **Win1251**, и в качестве типа *TCHAR* нужно применять *char*. Если объявлен макрос **UNICODE**, функция предполагает, что используются символы страницы 1049, и в качестве типа *TCHAR* нужно применять *short*.

Капитализация первой буквы влияет на анализ. На выход возвращаются слова, состоящие целиком из заглавных букв, в котором буква «Ё» заменена на «Е».

Остаток слова, превышающий константу *MO_MAXWORDLEN*, отсекается.

Рекомендуемый размер буфера *pszBuf* задается константой *MO_LARGEBUFSIZE*.

Если функция вернула 0, это означает, что морфологический анализ невозможен (например, введено слово «klop» или «*мама?»). В этом случае в буфер *pszBuf* помещается исходное слово.

См. также: [MoGetNormalForm](#), [MoGetAllNormalForms](#), [MoGetStem](#)

Функции для использования в приложениях анализа текста

Следующий набор функций позволяет провести полный морфологический анализ заданного слова и получить всю грамматическую информацию о нем. Данный набор функций доступен только в версии библиотеки **RCO Morphology Professional**.

- [MoAllocInfo](#)
- [MoFreeInfo](#)
- [MoAnalyze](#)
- Работа с [MoQueryInfo](#)

MoAllocInfo

Данная функция резервирует дескриптор результатов морфологического анализа.

```
void* MoAllocInfo(  
);
```

Возвращаемое значение

В случае успешного завершения функция возвращает дескриптор, иначе – *NULL*. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Linux**-версии в качестве кода ошибки используйте значение переменной `errno`.

См. также: [MoFreeInfo](#), [MoAnalyze](#), [MoQueryInfo](#)

MoFreeInfo

Данная функция освобождает ресурсы, связанные с дескриптором результатов морфологического анализа.

```
void MoFreeInfo(  
    void* hInfo // дескриптор  
);
```

Параметры

hInfo

[вх] Дескриптор, полученный в результате вызова **MoAllocInfo**.

См. также [MoAllocInfo](#), [MoAnalyze](#), [MoQueryInfo](#)

MoAnalyze

Данная функция производит полный морфологический анализ слова, после чего инициализирует дескриптор результатами анализа. Сначала предпринимается попытка точного морфологического анализа, в случае неудачи – анализа по правилам, при повторной неудаче – приближенного анализа.

```
int MoAnalyze(  
    void* hInfo, // дескриптор  
    TCHAR* pszWord, // анализируемое слово  
    unsigned uOptions // параметры алгоритма анализа  
);
```

Параметры

hInfo

[вх] Дескриптор, полученный в результате вызова **MoAllocInfo**.

pszWord

[вх] Слово для анализа. Указатель на буфер со строкой, завершающейся нулем.

uOptions

[вх] Параметры алгоритма при анализе. Комбинация следующих битовых флагов влияет на точный анализ и синтез:

<i>MO_O_LOWER_NONAME_NAME</i>	Слово, написанное с маленькой буквы, всегда может быть опознано как имя собственное.
<i>MO_O_LOWER_NONAME_NAME_NOHOMONIMS</i>	Слово, написанное с маленькой буквы, может быть опознано как имя собственное, только если оно не опознано как имя нарицательное.
<i>MO_O_LOWER_NONAME</i>	Слово, написанное с маленькой буквы, не может быть опознано как имя собственное.
<i>MO_O_UPPER_NAME_NONAME</i>	Слово, написанное с большой буквы, может быть опознано и как имя собственное, и как нарицательное.
<i>MO_O_UPPER_NAME</i>	Слово, написанное с большой буквы, не может быть опознано только как имя собственное.

Установка параметров *MO_O_LOWER_NONAME_NAME*, *MO_O_LOWER_NONAME_NAME_NOHOMONIMS*, *MO_O_LOWER_NONAME* является взаимоисключающей. То же относится к параметрам *MO_O_UPPER_NAME_NONAME* и *MO_O_UPPER_NAME*.

Если ни один из параметров не задан, используется изначальная маска, равная *MO_O_LOWER_NONAME_NAME_NOHOMONIMS* / *MO_O_UPPER_NAME_NONAME*.

Кроме того, допускается использование комбинации следующих битовых флагов, задающих параметры приближенного анализа:

<i>MO_O_NORMALFORM</i>	Предполагать, что слово стоит в нормальной форме.
<i>MO_O_NAME</i>	Предполагать, что слово является именем собственным.
<i>MO_O_NONAME</i>	Предполагать, что слово является именем нарицательным.
<i>MO_O_PLURAL_IMPOSSIBLE</i>	Предполагать, что слово стоит в форме единственного числа.
<i>MO_O_SUBJECT</i>	Искать в парадигме существительных.
<i>MO_O_DEFINITION</i>	Искать в парадигме прилагательных.
<i>MO_O_VERB</i>	Искать в парадигме глаголов.
<i>MO_O_UNCHANGED</i>	Искать в парадигме неизменяемых слов.

Если ни один из параметров не задан, то используется изначальная маска для значений, равная *MO_O_NAME* / *MO_O_NONAME* / *MO_O_SUBJECT* / *MO_O_DEFINITION* / *MO_O_VERB* / *MO_O_UNCHANGED*.

Возвращаемое значение

В случае успешного завершения функция возвращает *0*, иначе – *1*. В случае неудачи вызывайте **GetLastError** для получения кода ошибки. В **Linux**-версии в качестве кода ошибки используйте значение переменной `errno`.

Замечания.

Функция работает со строками в кодировке **Win1251**, и в качестве типа *TCHAR* нужно применять *char*. Если объявлен макрос **UNICODE**, функция предполагает, что используются символы страницы 1049, и в качестве типа *TCHAR* нужно применять *short*.

Капитализация первой буквы влияет на анализ.

Остаток слова, превышающий константу *MO_MAXWORDLEN*, отсекается.

В случае необходимости анализа нескольких слов подряд можно пользоваться одним дескриптором. При вызове функции результаты предыдущего анализа перекроются полученными результатами.

См. также: [MoAllocInfo](#), [MoFreeInfo](#), [MoQueryInfo](#)

MoQueryInfo

Данная функция позволяет получать информацию о результатах анализа слова функцией **MoAnalyze**.

```
int MoQueryInfo(
    void* hInfo, // дескриптор
    unsigned uInfoType, // тип требуемой информации
    int nParam1, // вспомогательный параметр 1
    int nParam2, // вспомогательный параметр 2
    void* pResult // указатель на результат
);
```

Параметры

hInfo

[вх] Дескриптор, полученный в результате вызова **MoAllocInfo**.

uInfoType

[вх] Тип запрашиваемой информации. Может принимать одно из следующих значений:

<i>MO_Q_ANALYSISTYPE</i>	Тип проведенного анализа.
<i>MO_Q_PARADIGMCOUNT</i>	Число вариантов разбора.
<i>MO_Q_WORDNORMALFORM</i>	Нормальная форма слова.
<i>MO_Q_WORDSTEM</i>	Основа слова.
<i>MO_Q_SUFFIXLEN</i>	Длина распознанного конца слова.
<i>MO_Q_PARTOFSPEECHCODE</i>	Код части речи.
<i>MO_Q_PARTOFSPEECHNAME</i>	Название части речи.
<i>MO_Q_ISANIMATE</i>	Признак одушевленности.
<i>MO_Q_FORMHITCOUNT</i>	Число вариантов грамматических форм слова.
<i>MO_Q_PARADIGMCODE</i>	Код парадигмы словоизменения.
<i>MO_Q_FORMHITINDEX</i>	Индекс грамматической формы в парадигме словоизменения.
<i>MO_Q_FORMCOUNT</i>	Максимально возможное число словоформ.
<i>MO_Q_FORMNAME</i>	Название грамматической формы.
<i>MO_Q_WORDFORM</i>	Заданная форма слова.

Значения и семантика остальных параметров зависят от типа запрашиваемой информации.

nParam1

[вх] Вспомогательный параметр.

nParam2

[вх] Вспомогательный параметр.

pResult

[вых] Указатель на переменную, которой будет присвоен результат запроса. В зависимости от запроса переменная должна иметь тип *TCHAR** или *int*. В первом случае строка, на которую указывает переменная, связана с дескриптором и будет удалена при вызове **MoFreeInfo**, а ее содержимое останется гарантировано неизменным до одного следующего вызова **MoAnalyze** или **MoQueryInfo**.

В случае неудачного завершения значение переменной, на которую указывает параметр *pResult*, будет неопределенным.

Возвращаемое значение

В случае успешного завершения функция возвращает *0*, иначе – *1*. При неудаче вызывайте **GetLastError** для получения кода ошибки. В **Linux**-версии в качестве кода ошибки используйте значение переменной `errno`.

Замечания.

Функция работает со строками в кодировке **Win1251**, и в качестве типа *TCHAR* нужно применять *char*. Если объявлен макрос **UNICODE**, функция предполагает, что используются символы страницы 1049, и в качестве типа *TCHAR* нужно применять *short*.

См. также: [MoAllocInfo](#), [MoFreeInfo](#), [MoAnalyze](#)

Работа с MoQueryInfo

В следующих разделах приведено описание возможных запросов к дескриптору результатов морфологического анализа посредством обращения к функции **MoQueryInfo**.

Тип проведенного анализа

Число вариантов разбора

Нормальная форма слова

Основа слова

Длина распознанного конца слова

Код части речи

Название части речи

Признак одушевленности

Число вариантов грамматических форм слова

Код парадигмы словоизменения

Индекс грамматической формы в парадигме словоизменения

Максимально возможное число словоформ

Название грамматической формы

Заданная форма слова

Тип проведенного анализа

Тип проведенного морфологического анализа.

```
int MoQueryInfo(  
    hInfo, // дескриптор  
    MO_Q_ANALYSISTYPE, // тип требуемой информации  
    0, // игнорируется, должен быть равен 0  
    0, // игнорируется, должен быть равен 0  
    &nResult // указатель на переменную типа int  
);
```

Параметры

nResult

[вых] Переменной *nResult* будет присвоено одно из следующих значений:

- 0 Удался точный анализ или анализ по правилам
 - 1 Удался приближенный анализ
 - 2 Анализ не удался
-

См. также: [MoQueryInfo](#)

Число вариантов разбора

Число вариантов разбора, полученных в результате морфологического анализа. Например, при анализе словоформы «стали» возникают два варианта, один из которых соответствует глаголу «стать», а другой – существительному «сталь».

```
int MoQueryInfo(  
    hInfo, // дескриптор  
    MO_Q_PARADIGMCOUNT, // тип требуемой информации  
    0, // игнорируется, должен быть равен 0  
    0, // игнорируется, должен быть равен 0  
    &nResult // указатель на переменную типа int  
);
```

Параметры

nResult

[вых] Переменной *nResult* будет присвоено одно из следующих значений:

- 0 Морфологический анализ не удался
 - 1 Точный анализ дал только один вариант разбора, либо проводился приближенный анализ
 - >1 Точный анализ дал несколько вариантов разбора
-

См. также: [MoQueryInfo](#)

Нормальная форма слова

Нормальная форма слова для заданного варианта разбора.

```
int MoQueryInfo(  
    hInfo, // дескриптор  
    MO_Q_WORDNORMALFORM, // тип требуемой информации  
    nParam1, // индекс варианта разбора  
    0, // игнорируется, должен быть равен 0  
    &pszResult // указатель на переменную типа TCHAR*  
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

pszResult

[вых] Переменной *pszResult* будет присвоен адрес буфера со строкой, завершающейся нулем и содержащей нормальную форму слова для данного варианта разбора.

См. также: [MoQueryInfo](#)

Основа слова

Основа слова для заданного варианта разбора.

```
int MoQueryInfo(  
    hInfo, // дескриптор  
    MO_Q_WORDSTEM, // тип требуемой информации  
    nParam1, // индекс варианта разбора  
    0, // игнорируется, должен быть равен 0  
    &pszResult // указатель на переменную типа TCHAR*  
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

pszResult

[вых] Переменной *pszResult* будет присвоен адрес буфера со строкой, завершающейся нулем и содержащей основу слова для данного варианта разбора.

См. также: [MoQueryInfo](#)

Длина распознанного конца слова

Длина распознанного конца слова в случае приближенного анализа.

```
int MoQueryInfo(  
    hInfo, // дескриптор  
    MO_Q_SUFFIXLEN, // тип требуемой информации  
    nParam1, // индекс варианта разбора  
    0, // игнорируется, должен быть равен 0  
    &nResult // указатель на переменную типа int  
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nResult

[вых] В случае приближенного анализа переменной *nResult* будет присвоена совместная длина окончания и суффикса, на основании которых было принято решение о способе словоизменения данного слова. Чем больше это значение – тем более достоверен результат приближенного анализа. В случае точного морфоанализа значение переменной *nResult* будет равно длине обработанной словоформы.

См. также: [MoQueryInfo](#)

Код части речи

Код части речи для заданного варианта разбора (см. [Приложение 2](#)).

```
int MoQueryInfo(  
    hInfo, // дескриптор  
    MO_Q_PARTOFSPEECHCODE, // тип требуемой информации  
    nParam1, // индекс варианта разбора  
    0, // игнорируется, должен быть равен 0  
    &nResult // указатель на переменную типа int  
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nResult

[вых] Переменной *nResult* будет присвоен код части речи. В случае приближенного анализа код будет равен [MO_POS_UNKNOWN](#), так как приближенный анализ осуществляется с точностью до парадигмы.

См. также: [MoQueryInfo](#)

Название части речи

Название части речи и/или лексико-семантического разряда для заданного варианта разбора. Сокращенные названия частей речи см. в [Приложении 2](#).

```
int MoQueryInfo(
    hInfo, // дескриптор
    MO_Q_PARTOFSPEECHNAME, // тип требуемой информации
    nParam1, // индекс варианта разбора
    0, // игнорируется, должен быть равен 0
    &pszResult // указатель на переменную типа TCHAR*
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

pszResult

[вых] Переменной *pszResult* будет присвоен адрес буфера со строкой, завершающейся нулем и содержащей название части речи для данного варианта разбора.

См. также: [MoQueryInfo](#)

Признак одушевленности

Признак одушевленности слова для заданного варианта разбора.

```
int MoQueryInfo(
    hInfo, // дескриптор
    MO_Q_ISANIMATE, // тип требуемой информации
    nParam1, // индекс варианта разбора
    0, // игнорируется, должен быть равен 0
    &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nResult

[вых] Переменной *nResult* будет присвоено одно из следующих значений:

- 0 Анализируемое слово либо не является существительным, либо является неодушевленным существительным
- 1 Анализируемое слово является одушевленным существительным
- 2 Анализируемое слово может выступать как в роли одушевленного, так и неодушевленного существительного

См. также: [MoQueryInfo](#)

Число вариантов грамматических форм слова

Число вариантов возможных грамматических форм слова для заданного варианта разбора. Например, для варианта разбора словоформы «стали», представляющего существительное «сталь», возможны грамматические формы родительного, дательного и предложного падежей единственного числа, а также формы именительного и винительного падежей множественного числа – всего пять вариантов.

```
int MoQueryInfo(  
    hInfo, // дескриптор  
    MO_Q_FORMNITCOUNT, // тип требуемой информации  
    nParam1, // индекс варианта разбора  
    0, // игнорируется, должен быть равен 0  
    &nResult // указатель на переменную типа int  
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nResult

[вых] Переменной *nResult* будет присвоена число вариантов грамматического описания слова. В случае приближенного анализа это число равно 0, в случае точного анализа – больше 0.

См. также: [MoQueryInfo](#)

Код парадигмы словоизменения

Код парадигмы словоизменения для заданного варианта разбора (см. [Приложение 1](#)).

```
int MoQueryInfo(  
    hInfo, // дескриптор  
    MO_Q_PARADIGMCODE, // тип требуемой информации  
    nParam1, // индекс варианта разбора  
    0, // игнорируется, должен быть равен 0  
    &nResult // указатель на переменную типа int  
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nResult

[вых] Переменной *nResult* будет присвоен код парадигмы.

См. также: [MoQueryInfo](#)

Индекс грамматической формы в парадигме словоизменения

Индекс грамматической формы в парадигме словоизменения. Позволяет определить грамматические характеристики словоформы на основе таблицы в [Приложении 1](#). Выбор таблицы зависит от типа парадигмы, который можно получить путем запроса [MO_Q_PARADIGMCODE](#).

```
int MoQueryInfo(
    hInfo, // дескриптор
    MO_Q_FORMHITINDEX, // тип требуемой информации
    nParam1, // индекс варианта разбора
    nParam2, // индекс варианта описания
    &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nParam2

[вх] Индекс варианта описания. Начинается с 0. Число вариантов описания можно получить путем запроса [MO_Q_FORMHITCOUNT](#).

nResult

[вых] Переменной *nResult* будет присвоен индекс грамматического описания в таблице словоизменения. Это значение может быть использовано в качестве параметра *nParam2* в запросах [MO_Q_FORMNAME](#) и [MO_Q_WORDFORM](#).

См. также: [MoQueryInfo](#)

Максимально возможное число словоформ

Потенциально возможное число словоформ для заданного варианта разбора. Определяет верхнюю границу диапазона значений, которые может принимать индекс грамматической формы. Используется для получения всех словоформ путем запроса [MO_Q_WORDFORM](#).

```
int MoQueryInfo(
    hInfo, // дескриптор
    MO_Q_FORMCOUNT, // тип требуемой информации
    nParam1, // индекс варианта разбора
    0, // игнорируется, должен быть равен 0
    &nResult // указатель на переменную типа int
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nResult

[вых] Переменной *nResult* будет присвоено число словоформ.

См. также: [MoQueryInfo](#)

Название грамматической формы

Название грамматической формы для заданного варианта разбора и индекса формы, который можно получить путем запроса [MO_Q_FORMHITINDEX](#).

```
int MoQueryInfo(
    hInfo, // дескриптор
    MO_Q_FORMNAME, // тип требуемой информации
    nParam1, // индекс варианта разбора
    nParam2, // индекс в таблице словоизменения
    &pszResult // указатель на переменную типа TCHAR*
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nParam2

[вх] Индекс в таблице словоизменения. Начинается с 0. Число вариантов описания можно получить путем запроса [MO_Q_FORMCOUNT](#).

nResult

[вых] Переменной *pszResult* будет присвоен адрес буфера со строкой, завершающейся нулем и содержащей название грамматической формы слова для данного варианта разбора и индекса в таблице словоизменения.

См. также: [MoQueryInfo](#)

Заданная форма слова

Построение словоформы для заданного варианта разбора и индекса грамматической формы в парадигме словоизменения (см. [Приложение 1](#)).

```
int MoQueryInfo(
    hInfo, // дескриптор
    MO_Q_WORDFORM, // тип требуемой информации
    nParam1, // индекс варианта разбора
    nParam2, // индекс в таблице словоизменения
    &pszResult // указатель на переменную типа TCHAR*
);
```

Параметры

nParam1

[вх] Индекс варианта разбора. Начинается с 0. Число вариантов разбора можно получить путем запроса [MO_Q_PARADIGMCOUNT](#).

nParam2

[вх] Индекс в таблице словоизменения. Начинается с 0. Число вариантов описания можно получить путем запроса [MO_Q_FORMCOUNT](#). Если индекс маскирован константой [MO_O_REFLEXIVE](#) ($nParam2 \mid MO_O_REFLEXIVE$), то для глаголов строится возвратная форма по индексу $nParam2 \& \sim MO_O_REFLEXIVE$.

nResult

[вых] Переменной *pszResult* будет присвоен адрес буфера со строкой, завершающейся нулем и содержащей построенную словоформу для данного варианта разбора и индекса в таблице словоизменения. В случае если существует несколько вариантов написания словоформы, все варианты помещаются в буфер и разделяются прямым слешем («/»), код 47). Если заданная форма у слова отсутствует, буфер будет содержать строку нулевой длины (состоящую из единственного символа ноль).

См. также: [MoQueryInfo](#)

Пример использования

В приведенном ниже примере иллюстрируется порядок работы с библиотекой, а также влияние некоторых настроек на результат анализа различных слов.

Исходный код тестового примера

```
#include <windows.h>
#include <stdio.h>
#include "gpmorph.h"

int SimpleTest( TCHAR* pszWord, unsigned uOptions, char* pOptionsText
    );

int FullTest( TCHAR* pszWord, unsigned uOptions, char* pOptionsText
    );

void main()
{
    TCHAR* s;

    s = TEXT( "лужков" );
    // распознавать имена собственные с маленькой буквы всегда
    SimpleTest( s, MO_O_LOWER_NONAME_NAME, "MO_O_LOWER_NONAME_NAME" );

    s = TEXT( "лужков" );
    // распознавать имена собственные с маленькой буквы только при
    // отсутствии других вариантов
    SimpleTest( s, MO_O_LOWER_NONAME_NAME_NOHOMONIMS,
        "MO_O_LOWER_NONAME_NAME_NOHOMONIMS" );
    // распознавать с большой буквы только имена собственные
    s = TEXT( "Лужков" );
    SimpleTest( s, MO_O_UPPER_NAME, "MO_O_UPPER_NAME" );

    // распознавать имена собственные с маленькой буквы только при
    // отсутствии других вариантов
    s = TEXT( "россия" );
    SimpleTest( s, MO_O_LOWER_NONAME_NAME_NOHOMONIMS,
        "MO_O_LOWER_NONAME_NAME_NOHOMONIMS" );

    // не распознавать имена собственные с маленькой буквы
    s = TEXT( "россия" );
    SimpleTest( s, MO_O_LOWER_NONAME_NAME_NOHOMONIMS,
        "MO_O_LOWER_NONAME_NAME_NOHOMONIMS" );

    // выбирать "лучший" вариант разбора, проверять гипотезу об имени
    // собственном
    // при написании с большой буквы
    // даже если слово есть в словаре как имя нарицательное
    s = TEXT( "Самолетов" );
    SimpleTest( s, MO_O_EXPANDBESTNORMALFORM |
        MO_O_OPTIMALQUERYPARSING, "MO_O_EXPANDBESTNORMALFORM |
        MO_O_OPTIMALQUERYPARSING" );

    s = TEXT( "Лужков" );
```

```

// распознавать с большой буквы все слова
FullTest( s, MO_O_UPPER_NAME_NONAME, "MO_O_UPPER_NAME_NONAME" );

s = TEXT( "Онтотротипилен" );
// не делать предположений о типе слова
FullTest( s, 0, "0" );

s = TEXT( "Онтотротипилен" );
// предполагать, что слово стоит в нормальной форме
FullTest( s, MO_O_NORMALFORM, "MO_O_NORMALFORM" );

s = TEXT( "Онтотротипилен" );
// предполагать, что слово является именем собственным
FullTest( s, MO_O_NAME, "MO_O_NAME" );

}

// работа с функциями для использования в ИПС
int SimpleTest( TCHAR* pszWord,
               unsigned uOptions, // параметры вызова функций
               char* pOptionsText // текст параметров вызова функций )
{
    TCHAR acSmallBuf[ MO_SMALLBUFSIZE ]; // буфер для результатов
    анализа
    TCHAR acLargeBuf[ MO_LARGEBUFSIZE ]; // буфер с результатами
    синтеза

    char *pszWordForm; // указатель для разбора списка словоформ
    int nBufSize, // размер исходного и возвращаемого буферов
        nRet; // результат вызова функции

    // загрузка словаря
    nRet = MoInitialize( NULL );
    if ( -1 == nRet )
    {
        printf( "Cannot initialize. Error=0x%8.8x\n", GetLastError() );
        return -1;
    }

    nBufSize = sizeof( acSmallBuf );
    nRet = MoGetAllNormalForms( pszWord, acSmallBuf, &nBufSize,
                               uOptions );
    printf( "MoGetAllNormalForms(%s, %s)=", pszWord, pOptionsText );
    for ( pszWordForm = acSmallBuf; *pszWordForm != 0; pszWordForm +=
        strlen(pszWordForm) + 1 )
        printf( "%s ", pszWordForm );
    printf( "\n" );

    nBufSize = sizeof( acSmallBuf );
    nRet = MoGetStem( pszWord, acSmallBuf, &nBufSize, uOptions );
    printf( "MoGetStem(%s, %s)=%s\n", pszWord, pOptionsText, acSmallBuf
        );

    nBufSize = sizeof( acLargeBuf );
    nRet = MoGetAllForms( pszWord, acLargeBuf, &nBufSize, uOptions );
    printf( "MoGetAllForms(%s, %s)=", pszWord, pOptionsText );

```

```
for ( pszWordForm = acLargeBuf; *pszWordForm != 0; pszWordForm +=
    strlen(pszWordForm) + 1 )
    printf( "%s ", pszWordForm );
printf( "\n" );

// выгружаем словарь
MoUninitialize();
return 0;
}

// работа с функциями для использования в лингвистическом ПО
int FullTest( TCHAR* pszWord,
    unsigned uOptions, // параметры вызова
    char* pOptionsText // текст параметров вызова функций )
{
    int nResult; // целочисленный результат запроса
    TCHAR *pszResult; // строковый результат запроса
    void *hInfo; // дескриптор результата разбора

    // вспомогательные переменные
    int nPgmCur, nPgmCount, nPgmCode, nFrmCur, nFrmCount, nRet;

    // загрузка словаря
    nRet = MoInitialize( NULL );
    if ( -1 == nRet )
    {
        printf( "Cannot initialize. Error=0x%8.8x\n", GetLastError() );
        return -1;
    }

    // резервируем дескриптор
    hInfo = MoAllocInfo();

    printf( "\nMoAnalyze(%s, %s)=\n", pszWord, pOptionsText );
    nRet = MoAnalyze( hInfo, pszWord, uOptions );

    // получаем результаты

    // тип анализа
    nRet = MoQueryInfo( hInfo, MO_Q_ANALYSISTYPE, 0, 0, &nResult );
    switch ( nResult )
    {
    case 0:
        printf( "Exact analysis performed\n" );
        break;
    case 1:
        printf( "Approximate analysis performed\n" );
        break;
    default:
        printf( "Cannot analyze\n" );
        goto failed;
    }

    // число вариантов
    nRet = MoQueryInfo( hInfo, MO_Q_PARADIGMCOUNT, 0, 0, &nPgmCount );

    for ( nPgmCur = 0; nPgmCur < nPgmCount; nPgmCur++ )
    {
```

```
// код парадигмы
nRet = MoQueryInfo( hInfo, MO_Q_PARADIGMCODE, nPgmCur, 0,
&nPgmCode );

// название парадигмы
nRet = MoQueryInfo( hInfo, MO_Q_PARADIGMNAME, nPgmCur, 0,
&pszResult );
printf( "Paradigm=%s\n", pszResult );

// название части речи
nRet = MoQueryInfo( hInfo, MO_Q_PARTOFSPEECHNAME, nPgmCur, 0,
&pszResult );
printf( "PartOfSpeech=%s\n", pszResult );

// нормальная форма
nRet = MoQueryInfo( hInfo, MO_Q_WORDNORMALFORM, nPgmCur, 0,
&pszResult );
printf( "NormalForm=%s\n", pszResult );

// основа слова
nRet = MoQueryInfo( hInfo, MO_Q_WORDSTEM, nPgmCur, 0, &pszResult
);
printf( "Stem=%s\n", pszResult );

// подходящие грамматические описания
printf( "RecognizedAs=" );
nRet = MoQueryInfo( hInfo, MO_Q_FORMHITCOUNT, nPgmCur, 0,
&nFrmCount );
for ( nFrmCur = 0; nFrmCur < nFrmCount; nFrmCur++ )
{
    nRet = MoQueryInfo( hInfo, MO_Q_FORMHITINDEX, nPgmCur, nFrmCur,
&nResult );
    nRet = MoQueryInfo( hInfo, MO_Q_FORMNAME, nPgmCur, nResult,
&pszResult );
    if ( 0 == nFrmCur ) printf( "%s", pszResult );
    else                printf( ", %s", pszResult );
}
printf( "\n" );

// все словоформы
printf( "AllForms=" );
nRet = MoQueryInfo( hInfo, MO_Q_FORMCOUNT, nPgmCur, 0, &nFrmCount
);
for ( nFrmCur = 0; nFrmCur < nFrmCount; nFrmCur++ )
{
    nRet = MoQueryInfo( hInfo, MO_Q_FORMNAME, nPgmCur, nFrmCur,
&pszResult );
    printf( "\n\t%s: ", pszResult );

    nRet = MoQueryInfo( hInfo, MO_Q_WORDFORM, nPgmCur, nFrmCur,
&pszResult );
    if ( *pszResult ) printf( "%s", pszResult );
    else                printf( "-" );

    // для глаголов строим возвратную форму
    if ( MO_PARADIGM_VERB == nPgmCode )
    {
```

```

        MoQueryInfo( hInfo, MO_Q_WORDFORM, nPgmCur, nFrmCur |
MO_O_REFLEXIVE, &pszResult );
        if ( *pszResult )
            printf( "/%s", pszResult );
    }
}

printf( "\n\n" );
}

failed:
// освобождение ресурсов
if ( hInfo ) MoFreeInfo( hInfo );
MoUninitialize();
return 0;
}

```

Результат работы тестового примера

```

MoGetAllNormalForms(лужков, MO_O_LOWER_NONAME_NAME)=ЛУЖКОВ ЛУЖОК
MoGetStem(лужков, MO_O_LOWER_NONAME_NAME)=ЛУЖК
MoGetAllForms(лужков, MO_O_LOWER_NONAME_NAME)=ЛУЖКА ЛУЖКАМ ЛУЖКАМИ ЛУЖКАХ
ЛУЖКЕ ЛУЖКИ ЛУЖКОВ ЛУЖКОВА ЛУЖКОВЕ ЛУЖКОВОЙ ЛУЖКОВУ ЛУЖКОВЫ ЛУЖКОВЫМ
ЛУЖКОВЫМИ ЛУЖКОВЫХ ЛУЖКОМ ЛУЖКУ ЛУЖОК
MoGetAllNormalForms(лужков, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=ЛУЖОК
MoGetStem(лужков, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=ЛУЖК
MoGetAllForms(лужков, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=ЛУЖКА ЛУЖКАМ
ЛУЖКАМИ ЛУЖКАХ ЛУЖКЕ ЛУЖКИ ЛУЖКОВ ЛУЖКОМ ЛУЖКУ ЛУЖОК
MoGetAllNormalForms(Лужков, MO_O_UPPER_NAME)=ЛУЖКОВ
MoGetStem(Лужков, MO_O_UPPER_NAME)=ЛУЖК
MoGetAllForms(Лужков, MO_O_UPPER_NAME)=ЛУЖКОВ ЛУЖКОВА ЛУЖКОВЕ ЛУЖКОВОЙ
ЛУЖКОВУ ЛУЖКОВЫ ЛУЖКОВЫМ ЛУЖКОВЫМИ ЛУЖКОВЫХ
MoGetAllNormalForms(россия, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=РОССИЯ
MoGetStem(россия, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=РОССИ
MoGetAllForms(россия, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=РОССИЕЙ РОССИЕЮ
РОССИИ РОССИЮ РОССИЯ
MoGetAllNormalForms(россия, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=РОССИЯ
MoGetStem(россия, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=РОССИ
MoGetAllForms(россия, MO_O_LOWER_NONAME_NAME_NOHOMONIMS)=РОССИЕЙ РОССИЕЮ
РОССИИ РОССИЮ РОССИЯ

MoAnalyze(Лужков, MO_O_UPPER_NAME_NONAME)=
Exact analysis perfomed
Paradigm=прилагательные
PartOfSpeech= фамилия
NormalForm=ЛУЖКОВ
Stem=ЛУЖКОВ
RecognizedAs=И. муж.
AllForms=
И. муж.: ЛУЖКОВ
Р. муж.: ЛУЖКОВА
Д. муж.: ЛУЖКОВУ
В. муж. одуш.: ЛУЖКОВА
В. муж. неодуш.: ЛУЖКОВА
Т. муж.: ЛУЖКОВЫМ
П. муж.: ЛУЖКОВЕ

```

кф. муж.: -
 И. жен.: ЛУЖКОВА
 Р. жен.: ЛУЖКОВОЙ
 Д. жен.: ЛУЖКОВОЙ
 В. жен.: ЛУЖКОВУ
 Т. жен.: ЛУЖКОВОЙ
 П. жен.: ЛУЖКОВОЙ
 кф. жен.: -
 И. ср.: -
 Р. ср.: -
 Д. ср.: -
 В. ср.: -
 Т. ср.: -
 П. ср.: -
 кф. ср.: -
 И. мн.: ЛУЖКОВЫ
 Р. мн.: ЛУЖКОВЫХ
 Д. мн.: ЛУЖКОВЫМ
 В. мн. одуш.: ЛУЖКОВЫХ
 В. мн. неодуш.: ЛУЖКОВЫХ
 Т. мн.: ЛУЖКОВЫМИ
 П. мн.: ЛУЖКОВЫХ
 кф. мн.: -
 сравн.: -

Paradigm=существительные

PartOfSpeech=Сущ. неодуш. муж.

NormalForm=ЛУЖОК

Stem=ЛУЖ

RecognizedAs=Р. мн.

AllForms=

И. ед.: ЛУЖОК
 Р. ед.: ЛУЖКА
 Д. ед.: ЛУЖКУ
 В. ед.: ЛУЖОК
 Т. ед.: ЛУЖКОМ
 П. ед.: ЛУЖКЕ
 Р2. ед.: -
 П2. ед.: ЛУЖКУ
 И. мн.: ЛУЖКИ
 Р. мн.: ЛУЖКОВ
 Д. мн.: ЛУЖКАМ
 В. мн.: ЛУЖКИ
 Т. мн.: ЛУЖКАМИ
 П. мн.: ЛУЖКАХ

MoAnalyze(Онтотротипилен, 0)=

Approximate analysis performed

Paradigm=существительные

PartOfSpeech=неизвестна

NormalForm=ОНТОТРОТИПИЛЬНЯ

Stem=ОНТОТРОТИПИЛ

RecognizedAs=Р. мн.

AllForms=

И. ед.: ОНТОТРОТИПИЛЬНЯ
 Р. ед.: ОНТОТРОТИПИЛЬНИ
 Д. ед.: ОНТОТРОТИПИЛЬНЕ
 В. ед.: ОНТОТРОТИПИЛЬНЮ
 Т. ед.: ОНТОТРОТИПИЛЬНЕЙ/ОНТОТРОТИПИЛЬНЕЮ

П. ед.: ОНТОТРОТИПИЛЬНЕ
 Р2. ед.: -
 П2. ед.: -
 И. мн.: ОНТОТРОТИПИЛЬНИ
 Р. мн.: ОНТОТРОТИПИЛЕН
 Д. мн.: ОНТОТРОТИПИЛЬНЯМ
 В. мн.: ОНТОТРОТИПИЛЬНИ
 Т. мн.: ОНТОТРОТИПИЛЬНЯМИ
 П. мн.: ОНТОТРОТИПИЛЬНЯХ

MoAnalyze(Онтотротипилен, МО_О_NORMALFORM)=

Approximate analysis performed

Paradigm=существительные

PartOfSpeech=неизвестна

NormalForm=ОНТОТРОТИПИЛЕН

Stem=ОНТОТРОТИПИЛЕН

RecognizedAs=И. ед., В. ед.

AllForms=

И. ед.: ОНТОТРОТИПИЛЕН
 Р. ед.: ОНТОТРОТИПИЛЕНА
 Д. ед.: ОНТОТРОТИПИЛЕНУ
 В. ед.: ОНТОТРОТИПИЛЕН
 Т. ед.: ОНТОТРОТИПИЛЕНОМ
 П. ед.: ОНТОТРОТИПИЛЕНЕ
 Р2. ед.: -
 П2. ед.: -
 И. мн.: ОНТОТРОТИПИЛЕНЫ
 Р. мн.: ОНТОТРОТИПИЛЕНОВ
 Д. мн.: ОНТОТРОТИПИЛЕНАМ
 В. мн.: ОНТОТРОТИПИЛЕНЫ
 Т. мн.: ОНТОТРОТИПИЛЕНАМИ
 П. мн.: ОНТОТРОТИПИЛЕНАХ

MoAnalyze(Онтотротипилен, МО_О_NAME)=

Approximate analysis performed

Paradigm=существительные

PartOfSpeech=неизвестна

NormalForm=ОНТОТРОТИПИЛЕН

Stem=ОНТОТРОТИПИЛЕН

RecognizedAs=И. ед., В. ед.

AllForms=

И. ед.: ОНТОТРОТИПИЛЕН
 Р. ед.: ОНТОТРОТИПИЛЕНА
 Д. ед.: ОНТОТРОТИПИЛЕНУ
 В. ед.: ОНТОТРОТИПИЛЕН
 Т. ед.: ОНТОТРОТИПИЛЕНОМ
 П. ед.: ОНТОТРОТИПИЛЕНЕ
 Р2. ед.: -
 П2. ед.: -
 И. мн.: -
 Р. мн.: -
 Д. мн.: -
 В. мн.: -
 Т. мн.: -
 П. мн.: -

Приложение 1.

Описание парадигмы словоизменения

Парадигма словоизменения представляет собой множество словоформ, которые может принимать слово в определенных грамматических формах. В зависимости от части речи группы слов могут потенциально иметь ограниченный набор грамматических форм. Вследствие этого все парадигмы разделяются на классы таким образом, что каждый класс парадигм описывает изменение слов, относящихся к одному грамматическому разряду.

В принятой здесь системе описания словоизменения существуют три полноценных класса парадигм: существительного, прилагательного и глагола. К четвертому, вырожденному классу, относятся неизменяемые слова. Словоизменение всех прочих частей речи (см. [Приложение 2](#)) строится по парадигмам одного из этих классов. Причастия не рассматриваются как самостоятельная часть речи, и все их формы включены в парадигму глагола.

Описание кодов парадигм.

<i>MO_PARADIGM_SUBJECT</i>	класс существительных
<i>MO_PARADIGM_DEFINITION</i>	класс прилагательных
<i>MO_PARADIGM_VERB</i>	класс глаголов
<i>MO_PARADIGM_UNCHANGED</i>	неизменяемые слова

Ниже приведен порядок следования грамматических форм в парадигмах словоизменения.

Слова, изменяющиеся по типу существительного (код *MO_PARADIGM_SUBJECT*), могут иметь формы, указанные в таблице ниже.

Индекс	Грамматическая форма	Пример
0	имен. падеж, един.	река
1	род. падеж, един.	реки
2	дат. падеж, един.	реке
3	вин. падеж, един.	реку
4	твор. падеж, един.	рекой, рекою
5	пред. падеж, един.	реке
6	2-ой. род. падеж, един.	-
7	2-ой. пред. падеж, един.	-
8	им. падеж, множ.	реки
9	род. падеж, множ.	рек
10	дат. падеж, множ.	рекам
11	вин. падеж, множ.	реки
12	твор. падеж, множ.	реками
13	пред. падеж, множ.	реках

Слова, изменяющиеся по типу прилагательного (код *MO_PARADIGM_DEFINITION*), могут иметь формы, указанные в таблице ниже.

Индекс	Грамматическая форма	Пример
0	имен. падеж, един., муж.	быстрый
1	род. падеж, един., муж.	быстрого
2	дат. падеж, един., муж.	быстрому
3	вин. падеж, един., муж., одуш.	быстрого
4	вин. падеж, един., муж., неодуш.	быстрый
5	твор. падеж, един., муж.	быстрым
6	пред. падеж, един., муж.	быстром
7	краткая форма, муж.	быстр
8	имен. падеж, един., жен.	быстрая
9	род. падеж, един., жен.	быстрой
10	дат. падеж, един., жен.	быстрой
11	вин. падеж, един., жен.	быструю
12	твор. падеж, един., жен.	быстрой, быстрою
13	пред. падеж, един., жен.	быстрой
14	краткая форма, жен.	быстра
15	имен. падеж, един., сред.	быстрое
16	род. падеж, един., сред.	быстрого
17	дат. падеж, един., сред.	быстрому
18	вин. падеж, един., сред.	быстрое
19	твор. падеж, един., сред.	быстрым
20	пред. падеж, един., сред.	быстром
21	краткая форма, сред.	быстро
22	имен. падеж, множ.	быстрые
23	род. падеж, множ.	быстрых
24	дат. падеж, множ.	быстрым
25	вин. падеж, множ., одуш.	быстрых
26	вин. падеж, множ., неодуш.	быстрые
27	твор. падеж, множ.	быстрыми
28	пред. падеж, множ.	быстрых
29	краткая форма, множ.	быстры
30	сравнительная степень	быстрее, быстреей

Слова, изменяющиеся по типу глагола (код *MO_PARADIGM_VERB*), могут иметь формы, указанные в таблице ниже.

Индекс	Грамматическая форма	Пример
0	Инф.	перифотографировать
1	Возв. инф.	перифотографироваться
2	Повел. Накл.	перифотографируй
3	Множ. повел.	перифотографируйте
4	Буд. 1-е ед.	перифотографирую
5	Буд. 1-е мн.	перифотографируем
6	Буд. 2-е ед.	перифотографируешь
7	Буд. 2-е мн.	перифотографируете
8	Буд. 3-е ед.	перифотографирует
9	Буд. 3-е мн.	перифотографируют
10	Наст. 1-е ед.	перифотографирую
11	Наст. 1-е мн.	перифотографируем
12	Наст. 2-е ед.	перифотографируешь
13	Наст. 2-е мн.	перифотографируете
14	Наст. 3-е ед.	перифотографирует

15	Наст. 3-е мн.	перифотографируют
16	Наст. прич. действ. И. ед.	перифотографирующий
17	Наст. прич. страд. И. ед.	перифотографируемый
18	Наст. дееприч.	перифотографируя
19	Прош. муж.	перифотографировал
20	Прош. жен.	перифотографировала
21	Прош. ср.	перифотографировало
22	Прош. множ.	перифотографировали
23	Прош. прич. действ. И. ед.	перифотографировавший
24	Прош. прич. страд. И. ед.	перифотографированный
25	Прош. дееприч.	перифотографировав, перифотографировавши
26	Наст. прич. действ. Р. муж.	перифотографирующего
27	Наст. прич. действ. Д. муж.	перифотографирующему
28	Наст. прич. действ. В. муж. одуш.	перифотографирующего
29	Наст. прич. действ. В. муж. неодуш.	перифотографирующий
30	Наст. прич. действ. Т. муж.	перифотографирующим
31	Наст. прич. действ. П. муж.	перифотографирующем
32	Наст. прич. действ. кф. муж.	-
33	Наст. прич. действ. И. жен.	перифотографирующая
34	Наст. прич. действ. Р. жен.	перифотографирующей
35	Наст. прич. действ. Д. жен.	перифотографирующей
36	Наст. прич. действ. В. жен.	перифотографирующую
37	Наст. прич. действ. Т.1 жен.	перифотографирующей
38	Наст. прич. действ. Т.2 жен.	перифотографирующей
39	Наст. прич. действ. П. жен.	перифотографирующей
40	Наст. прич. действ. кф. жен.	-
41	Наст. прич. действ. И. ср.	перифотографирующее
42	Наст. прич. действ. Р. ср.	перифотографирующего
43	Наст. прич. действ. Д. ср.	перифотографирующему
44	Наст. прич. действ. В. ср.	перифотографирующее
45	Наст. прич. действ. Т. ср.	перифотографирующим
46	Наст. прич. действ. П. ср.	перифотографирующем
47	Наст. прич. действ. кф. ср.	-
48	Наст. прич. действ. И. мн.	перифотографирующие
49	Наст. прич. действ. Р. мн.	перифотографирующих
50	Наст. прич. действ. Д. мн.	перифотографирующим
51	Наст. прич. действ. В. мн. одуш.	перифотографирующих
52	Наст. прич. действ. В. мн. неодуш.	перифотографирующие
53	Наст. прич. действ. Т. мн.	перифотографирующими
54	Наст. прич. действ. П. мн.	перифотографирующих
55	Наст. прич. действ. кф. мн.	-
56	Прош. прич. действ. Р. муж.	перифотографировавшего
57	Прош. прич. действ. Д. муж.	перифотографировавшему
58	Прош. прич. действ. В. муж. одуш.	перифотографировавшего
59	Прош. прич. действ. В. муж. неодуш.	перифотографировавший
60	Прош. прич. действ. Т. муж.	перифотографировавшим
61	Прош. прич. действ. П. муж.	перифотографировавшем
62	Прош. прич. действ. кф. муж.	-
63	Прош. прич. действ. И. жен.	перифотографировавшая

64	Прош. прич. действ. Р. жен.	перифотографировавшей
65	Прош. прич. действ. Д. жен.	перифотографировавшей
65	Прош. прич. действ. В. жен.	перифотографировавшую
67	Прош. прич. действ. Т.1 жен.	перифотографировавшей
68	Прош. прич. действ. Т.2 жен.	перифотографировавшую
69	Прош. прич. действ. П. жен.	перифотографировавшей
70	Прош. прич. действ. кф. жен.	-
71	Прош. прич. действ. И. ср.	перифотографировавшее
72	Прош. прич. действ. Р. ср.	перифотографировавшего
73	Прош. прич. действ. Д. ср.	перифотографировавшему
74	Прош. прич. действ. В. ср.	перифотографировавшее
75	Прош. прич. действ. Т. ср.	перифотографировавшим
76	Прош. прич. действ. П. ср.	перифотографировавшем
77	Прош. прич. действ. кф. ср.	-
78	Прош. прич. действ. И. мн.	перифотографировавшие
79	Прош. прич. действ. Р. мн.	перифотографировавших
80	Прош. прич. действ. Д. мн.	перифотографировавшим
81	Прош. прич. действ. В. мн. одуш.	перифотографировавших
82	Прош. прич. действ. В. мн. неодуш.	перифотографировавшие
83	Прош. прич. действ. Т. мн.	перифотографировавшими
84	Прош. прич. действ. П. мн.	перифотографировавших
85	Прош. прич. действ. кф. мн.	-
86	Наст. прич. страд. Р. муж.	перифотируемого
87	Наст. прич. страд. Д. муж.	перифотируемому
88	Наст. прич. страд. В. муж. одуш.	перифотируемого
89	Наст. прич. страд. В. муж. неодуш.	перифотируемый
90	Наст. прич. страд. Т. муж.	перифотируемым
91	Наст. прич. страд. П. муж.	перифотируемом
92	Наст. прич. страд. кф. муж.	перифотируем
93	Наст. прич. страд. И. жен.	перифотируемая
94	Наст. прич. страд. Р. жен.	перифотируемой
95	Наст. прич. страд. Д. жен.	перифотируемой
96	Наст. прич. страд. В. жен.	перифотируемую
97	Наст. прич. страд. Т.1 жен.	перифотируемой
98	Наст. прич. страд. Т.2 жен.	перифотируемую
99	Наст. прич. страд. П. жен.	перифотируемой
100	Наст. прич. страд. кф. жен.	перифотируема
101	Наст. прич. страд. И. ср.	перифотируемое
102	Наст. прич. страд. Р. ср.	перифотируемого
103	Наст. прич. страд. Д. ср.	перифотируемому
104	Наст. прич. страд. В. ср.	перифотируемое
105	Наст. прич. страд. Т. ср.	перифотируемым
106	Наст. прич. страд. П. ср.	перифотируемом
107	Наст. прич. страд. кф. ср.	перифотируемо
108	Наст. прич. страд. И. мн.	перифотируемые
109	Наст. прич. страд. Р. мн.	перифотируемых
110	Наст. прич. страд. Д. мн.	перифотируемым
111	Наст. прич. страд. В. мн. одуш.	перифотируемых
112	Наст. прич. страд. В. мн. неодуш.	перифотируемые
113	Наст. прич. страд. Т. мн.	перифотируемыми

114	Наст. прич. страд. П. мн.	перифотографируемых
115	Наст. прич. страд. кф. мн.	перифотографируемы
116	Прош. прич. страд. Р. муж.	перифотографированного
117	Прош. прич. страд. Д. муж.	перифотографированному
118	Прош. прич. страд. В. муж. одуш.	перифотографированного
119	Прош. прич. страд. В. муж. неодуш.	перифотографированный
120	Прош. прич. страд. Т. муж.	перифотографированным
121	Прош. прич. страд. П. муж.	перифотографированном
122	Прош. прич. страд. кф. муж.	перифотографирован
123	Прош. прич. страд. И. жен.	перифотографированная
124	Прош. прич. страд. Р. жен.	перифотографированной
125	Прош. прич. страд. Д. жен.	перифотографированной
126	Прош. прич. страд. В. жен.	перифотографированную
127	Прош. прич. страд. Т.1 жен.	перифотографированной
128	Прош. прич. страд. Т.2 жен.	перифотографированную
129	Прош. прич. страд. П. жен.	перифотографированной
130	Прош. прич. страд. кф. жен.	перифотографирована
131	Прош. прич. страд. И. ср.	перифотографированное
132	Прош. прич. страд. Р. ср.	перифотографированного
133	Прош. прич. страд. Д. ср.	перифотографированному
134	Прош. прич. страд. В. ср.	перифотографированное
135	Прош. прич. страд. Т. ср.	перифотографированным
136	Прош. прич. страд. П. ср.	перифотографированном
137	Прош. прич. страд. кф. ср.	перифотографировано
138	Прош. прич. страд. И. мн.	перифотографированные
139	Прош. прич. страд. Р. мн.	перифотографированных
140	Прош. прич. страд. Д. мн.	перифотографированным
141	Прош. прич. страд. В. мн. одуш.	перифотографированных
142	Прош. прич. страд. В. мн. неодуш.	перифотографированные
143	Прош. прич. страд. Т. мн.	перифотографированными
144	Прош. прич. страд. П. мн.	перифотографированных
145	Прош. прич. страд. кф. мн.	перифотографированы

Приложение 2.

Классификация частей речи

Ниже приведены коды частей речи (лексико-грамматических разрядов).

В последнем столбце указан класс парадигмы словоизменения, который определяет возможные грамматические формы слов (см. [Приложение 1](#)):

Н – неизменяемое, С – существительное, П – прилагательное, Г – глагол.

Часть речи	Код	Пример	Класс
неизвестна	<i>MO_POS_UNKNOWN</i>		
Вводное слово	<i>MO_POS_INTRODUCTORY</i>	просто	Н
Междометье	<i>MO_POS_INTERJECTION</i>	ах	Н
Предикатив	<i>MO_POS_PREDICATIVE</i>	просто	Н
Предлог	<i>MO_POS_PREPOSITION1</i>	у	Н
Предлог	<i>MO_POS_PREPOSITION2</i>	к	Н
Предлог	<i>MO_POS_PREPOSITION3</i>	в	Н
Предлог	<i>MO_POS_PREPOSITION4</i>	с	Н
Предлог	<i>MO_POS_PREPOSITION5</i>	на	Н
Союз сочинительный	<i>MO_POS_CONJUNCTION COORDINATIVE</i>	и	Н
Союз	<i>MO_POS_CONJUNCTION1</i>	что	Н
Союз	<i>MO_POS_CONJUNCTION2</i>	что	Н
Союз	<i>MO_POS_CONJUNCTION3</i>	что	Н
Частица	<i>MO_POS_PARTICLE</i>	же	Н
Наречие	<i>MO_POS_ADVERB</i>	просто	Н
Сущ. сокращенное	<i>MO_POS_ABBREVIATEDNOUN</i>	выкл.	Н
Прил. сокращенное	<i>MO_POS_ABBREVIATEDADJECTIVE</i>	др.	Н
Сокращенное вводное слово	<i>MO_POS_ABBREVIATED INTRODUCTORY</i>	напр.	Н
Обособленная сравнительная степень	<i>MO_POS_COMPARATIVE</i>	побольше	Н
Притяжательное прилагательное	<i>MO_POS_POSSESSIVEADJECTIVE</i>	птичий	П
Мест. муж.	<i>MO_POS_PRONOUN</i>	он	С
Мест. жен.	<i>MO_POS_PRONOUNFEMININE</i>	она	С
Мест. сред.	<i>MO_POS_PRONOUNNEUTER</i>	оно	С
Местоименное прилагательное	<i>MO_POS_PRONOUNADJECTIVE</i>	такой	П
Числ.	<i>MO_POS_NUMERIC</i>	три	С
Собирательное числ.	<i>MO_POS_NUMERICCOLLECTIVE</i>	трое	С
Числ. порядковое	<i>MO_POS_NUMERICORDINAL</i>	третий	П
Аббр. муж	<i>MO_POS_ABBREVIATURE</i>	ВАК	Н
Аббр. жен.	<i>MO_POS_ABBREVIATUREFEMININE</i>	РФ	Н
Аббр. ср.	<i>MO_POS_ABBREVIATURENEUTER</i>	ЦРУ	Н
Имя собств. муж.	<i>MO_POS_NAMEMASCULINE</i>	Саша	С
Имя собств. жен.	<i>MO_POS_NAMEFEMININE</i>	Мария	С
Отчество муж.	<i>MO_POS_MIDDLENAMEMASCULINE</i>	Васильевич	С

Отчество жен.	<i>MO_POS_MIDDLENAMEFEMININE</i>	Васильевна	С
Фамилия	<i>MO_POS_LASTNAME</i>	Иванов	П
Географ. название муж.	<i>MO_POS_GEOGRAPHYNAME MASCULINE</i>	Орел	С
Географ. название жен.	<i>MO_POS_GEOGRAPHYNAMEFEMININE</i>	Обь	С
Географ. название сред.	<i>MO_POS_GEOGRAPHYNAME NEUTER</i>	Внуково	С
Географ. название множ.	<i>MO_POS_GEOGRAPHYNAME PLURAL</i>	Карпаты	С
Глагол переходный несов. вида	<i>MO_POS_VERB IMPERFECT</i>	смотреть	Г
Глагол непереходный несов. вида	<i>MO_POS_VERB INTRANSITIVE IMPERFECT</i>	чихать	Г
Глагол переходный совершенного вида	<i>MO_POS_VERB PERFECT</i>	посмотреть	Г
Глагол непереходный совершенного вида	<i>MO_POS_VERB INTRANSITIVE PERFECT</i>	чихнуть	Г
Глагол переходный двувидовой	<i>MO_POS_VERB TOWAY</i>	эмитировать	Г
Глагол непереходный двувидовой	<i>MO_POS_VERB INTRANSITIVE TOWAY</i>	эмигрировать	Г
Прил.	<i>MO_POS_ADJECTIVE 1</i>	<u>фонетический</u>	П
Прил.	<i>MO_POS_ADJECTIVE 2</i>	<u>фонетический</u>	П
Сущ. неодуш. муж.	<i>MO_POS_NOUN INANIMATE MASCULINE</i>	трактор	С
Сущ. одуш. муж.	<i>MO_POS_NOUN ANIMATE MASCULINE</i>	зяблик	С
Сущ. одуш.-неодуш. муж.	<i>MO_POS_NOUN ANIMATE INANIMATE MASCULINE</i>	<u>кодировщик</u>	С
Сущ. неодуш. жен.	<i>MO_POS_NOUN INANIMATE FEMININE</i>	весна	С
Сущ. одуш. жен.	<i>MO_POS_NOUN ANIMATE FEMININE</i>	птичка	С
Сущ. одуш.-неодуш. жен.	<i>MO_POS_NOUN ANIMATE INANIMATE FEMININE</i>	матрешка	С
Сущ. неодуш. сред.	<i>MO_POS_NOUN INANIMATE NEUTER</i>	озеро	С
Сущ. одуш. сред.	<i>MO_POS_NOUN ANIMATE NEUTER</i>	дитя	С
Сущ. одуш.-неодуш. сред.	<i>MO_POS_NOUN ANIMATE INANIMATE NEUTER</i>	детище	С
Сущ. неодуш. множ.	<i>MO_POS_NOUN INANIMATE PLURAL</i>	ножницы	С
Наименование муж. рода	<i>MO_POS_APPELATION MASCULINE</i>	Гарант-Парк-интернет	С
Наименование жен. рода	<i>MO_POS_APPELATION FEMININE</i>	Формоза	С
Наименование сред. рода	<i>MO_POS_APPELATION NEUTER</i>	Алеко	С
Наименование множ. числа	<i>MO_POS_APPELATION PLURAL</i>	Инфо-Технологии	С