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.

Содержание

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 вызваться 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 отобрать 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 удается 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:

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 и как имя собственное, и как нарицательное.
MO_O_UPPER_NAME	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 удается 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:

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.
MO_O_SUBJECT	Искать в парадигме существительных.
MO_O_DEFINITION	Искать в парадигме прилагательных.
MO_O_VERB	Искать в парадигме глаголов.
MO_O_UNCHANGED	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:

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 и как имя собственное, и как нарицательное.
MO_O_UPPER_NAME	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*:

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_SUBJECT	Искать в парадигме существительных.
MO_O_DEFINITION	Искать в парадигме прилагательных.
MO_O_VERB	Искать в парадигме глаголов.
MO_O_UNCHANGED	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:

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 и как имя собственное, и как нарицательное.
MO_O_UPPER_NAME	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:

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.
MO_O_SUBJECT	Искать в парадигме существительных.
MO_O_DEFINITION	Искать в парадигме прилагательных.
MO_O_VERB	Искать в парадигме глаголов.
MO_O_UNCHANGED	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:

MO_Q_ANALYSISTYPE	Тип проведенного анализа.
MO_Q_PARADIGMCOUNT	Число вариантов разбора.
MO_Q_WORDNORMALFORM	Нормальная форма слова.
MO_Q_WORDSTEM	Основа слова.
MO_Q_SUFFIXLEN	Длина распознанного конца слова.
MO_Q_PARTOFSPEECHCODE	Код части речи.
MO_Q_PARTOFSPEECHNAME	Название части речи.
MO_Q_ISANIMATE	Признак одушевленности.
MO_Q_FORMHITCOUNT	Число вариантов грамматических форм слова.
MO_Q_PARADIGMCODE	Код парадигмы словоизменения.
MO_Q_FORMHITINDEX	Индекс грамматической формы в парадигме словоизменения.
MO_Q_FORMCOUNT	Максимально возможное число словоформ.
MO_Q_FORMNAME	Название грамматической формы.
MO_Q_WORDFORM	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_FORMHITCOUNT, // 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* | *MO_O_REFLEXIVE*), то для глаголов строится возвратная форма по индексу *nParam2* & *~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( "Excact analysis perfomed\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)=
Excact 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, MO_O_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, MO_O_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.

MO_PARADIGM_SUBJECT	класс существительных
MO_PARADIGM_DEFINITION	класс прилагательных
MO_PARADIGM_VERB	класс глаголов
MO_PARADIGM_UNCHANGED	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, быстрей

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

Индекс	Грамматическая форма	Пример
0	Инф.	перефотографировать
1	Возв. инф.	перефотографироваться
2	Повел. Накл.	перефотографируй
3	Множ. повел.	перефотографируйте
4	Буд. 1-е ед.	перефотографирую
5	Буд. 1-е мн.	перефотографируем
6	Буд. 2-е ед.	перефотографируешь
7	Буд. 2-е мн.	перефотографируете
8	Буд. 3-е ед.	перефотографирует
9	Буд. 3-е мн.	перефотографируют
10	Наст. 1-е ед.	перефотографирую
11	Наст. 1-е мн.	перефотографируем
12	Наст. 2-е ед.	перефотографируешь
13	Наст. 2-е мн.	перефотографируете
14	Наст. 3-е ед.	перефотографирует

15	Наст. 3-е мн.	перефотографируют
16	Наст. прич. действ. И. ед.	перефотографирующий
17	Наст. прич. страд. И. ед.	перефотографируемый
18	Наст. дееприч.	перефотографируя
19	Прош. муж.	перефотографировал
20	Прош. жен.	перефотографировала
21	Прош. ср.	перефотографировало
22	Прош. множ.	перефотографировали
23	Прош. прич. действ. И. ед.	перефотографировавший
24	Прош. прич. страд. И. ед.	перефотографированный
25	Прош. дееприч.	перефотографировав, перефотографировавши
26	Наст. прич. действ. Р. муж.	перефотографирующего
27	Наст. прич. действ. Д. муж.	перефотографирующему
28	Наст. прич. действ. В. муж. одуш.	перефотографирующего
29	Наст. прич. действ. В. муж. неодуш.	перефотографирующий
30	Наст. прич. действ. Т. муж.	перефотографирующим
31	Наст. прич. действ. П. муж.	перефотографирующем
32	Наст. прич. действ. кф. муж.	-
33	Наст. прич. действ. И. жен.	перефотографирующая
34	Наст. прич. действ. Р. жен.	перефотографирующей
35	Наст. прич. действ. Д. жен	перефотографирующей
36	Наст. прич. действ. В. жен.	перефотографирующую
37	Наст. прич. действ. Т.1 жен.	перефотографирующей
38	Наст. прич. действ. Т.2 жен.	перефотографирующею
39	Наст. прич. действ. П. жен.	перефотографирующей
40	Наст. прич. действ. кф. жен.	-
41	Наст. прич. действ. И. ср.	перефотографирующее
42	Наст. прич. действ. Р. ср.	перефотографирующего
43	Наст. прич. действ. Д. ср.	перефотографирующему
44	Наст. прич. действ. В. ср.	перефотографирующее
45	Наст. прич. действ. Т. ср.	перефотографирующим
46	Наст. прич. действ. П. ср.	перефотографирующем
47	Наст. прич. действ. кф. ср.	-
48	Наст. прич. действ. И. мн.	перефотографирующие
49	Наст. прич. действ. Р. мн.	перефотографирующих
50	Наст. прич. действ. Д. мн.	перефотографирующим
51	Наст. прич. действ. В. мн. одуш.	перефотографирующих
52	Наст. прич. действ. В. мн. неодуш.	перефотографирующие
53	Наст. прич. действ. Т. мн.	перефотографирующими
54	Наст. прич. действ. П. мн.	перефотографирующих
55	Наст. прич. действ. кф. мн.	-
56	Прош. прич. действ. Р. муж.	перефотографировавшего
57	Прош. прич. действ. Д. муж.	перефотографировавшему
58	Прош. прич. действ. В. муж. одуш.	перефотографировавшего
59	Прош. прич. действ. В. муж. неодуш.	перефотографировавший
60	Прош. прич. действ. Т. муж.	перефотографировавшим
61	Прош. прич. действ. П. муж.	перефотографировавшем
62	Прош. прич. действ. кф. муж.	-
63	Прош. прич. действ. И. жен.	перефотографировавшая

64	Прош. прич. действ. Р. жен.	перефотографировавшей
65	Прош. прич. действ. Д. жен.	перефотографировавшей
65	Прош. прич. действ. В. жен.	перефотографировавшую
67	Прош. прич. действ. Т.1 жен.	перефотографировавшей
68	Прош. прич. действ. Т.2 жен.	перефотографировавшею
69	Прош. прич. действ. П. жен.	перефотографировавшей
70	Прош. прич. действ. кф. жен.	-
71	Прош. прич. действ. И. ср.	перефотографировавшее
72	Прош. прич. действ. Р. ср.	перефотографировавшего
73	Прош. прич. действ. Д. ср.	перефотографировавшему
74	Прош. прич. действ. В. ср.	перефотографировавшее
75	Прош. прич. действ. Т. ср.	перефотографировавшим
76	Прош. прич. действ. П. ср.	перефотографировавшем
77	Прош. прич. действ. кф. ср.	-
78	Прош. прич. действ. И. мн.	перефотографировавшие
79	Прош. прич. действ. Р. мн.	перефотографировавших
80	Прош. прич. действ. Д. мн.	перефотографировавшим
81	Прош. прич. действ. В. мн. одуш.	перефотографировавших
82	Прош. прич. действ. В. мн. неодуш.	перефотографировавшие
83	Прош. прич. действ. Т. мн.	перефотографировавшими
84	Прош. прич. действ. П. мн.	перефотографировавших
85	Прош. прич. действ. кф. мн.	-
86	Наст. прич. страд. Р. муж.	перефотографируемого
87	Наст. прич. страд. Д. муж.	перефотографируемому
88	Наст. прич. страд. В. муж. одуш.	перефотографируемого
89	Наст. прич. страд. В. муж. неодуш.	перефотографируемый
90	Наст. прич. страд. Т. муж.	перефотографируемым
91	Наст. прич. страд. П. муж.	перефотографируемом
92	Наст. прич. страд. кф. муж.	перефотографируем
93	Наст. прич. страд. И. жен.	перефотографируемая
94	Наст. прич. страд. Р. жен.	перефотографируемой
95	Наст. прич. страд. Д. жен.	перефотографируемой
96	Наст. прич. страд. В. жен.	перефотографируемую
97	Наст. прич. страд. Т.1 жен.	перефотографируемой
98	Наст. прич. страд. Т.2 жен.	перефотографируемою
99	Наст. прич. страд. П. жен.	перефотографируемой
100	Наст. прич. страд. кф. жен.	перефотографируема
101	Наст. прич. страд. И. ср.	перефотографируемое
102	Наст. прич. страд. Р. ср.	перефотографируемого
103	Наст. прич. страд. Д. ср.	перефотографируемому
104	Наст. прич. страд. В. ср.	перефотографируемое
105	Наст. прич. страд. Т. ср.	перефотографируемым
106	Наст. прич. страд. П. ср.	перефотографируемом
107	Наст. прич. страд. кф. ср.	перефотографируемо
108	Наст. прич. страд. И. мн.	перефотографируемые
109	Наст. прич. страд. Р. мн.	перефотографируемых
110	Наст. прич. страд. Д. мн.	перефотографируемым
111	Наст. прич. страд. В. мн. одуш.	перефотографируемых
112	Наст. прич. страд. В. мн. неодуш.	перефотографируемые
113	Наст. прич. страд. Т. мн.	перефотографируемыми

114	Наст. прич. страд. П. мн.	перефотографируемых
115	Наст. прич. страд. кф. мн.	перефотографируемы
116	Прош. прич. страд. Р. муж.	перефотографированного
117	Прош. прич. страд. Д. муж.	перефотографированному
118	Прош. прич. страд. В. муж. одуш.	перефотографированного
119	Прош. прич. страд. В. муж. неодуш.	перефотографированный
120	Прош. прич. страд. Т. муж.	перефотографированным
121	Прош. прич. страд. П. муж.	перефотографированном
122	Прош. прич. страд. кф. муж.	перефотографирован
123	Прош. прич. страд. И. жен.	перефотографированная
124	Прош. прич. страд. Р. жен.	перефотографированной
125	Прош. прич. страд. Д. жен.	перефотографированной
126	Прош. прич. страд. В. жен.	перефотографированную
127	Прош. прич. страд. Т.1 жен.	перефотографированной
128	Прош. прич. страд. Т.2 жен.	перефотографированною
129	Прош. прич. страд. П. жен.	перефотографированной
130	Прош. прич. страд. кф. жен.	перефотографирована
131	Прош. прич. страд. И. ср.	перефотографированное
132	Прош. прич. страд. Р. ср.	перефотографированного
133	Прош. прич. страд. Д. ср.	перефотографированному
134	Прош. прич. страд. В. ср.	перефотографированное
135	Прош. прич. страд. Т. ср.	перефотографированным
136	Прош. прич. страд. П. ср.	перефотографированном
137	Прош. прич. страд. кф. ср.	перефотографировано
138	Прош. прич. страд. И. мн.	перефотографированные
139	Прош. прич. страд. Р. мн.	перефотографированных
140	Прош. прич. страд. Д. мн.	перефотографированным
141	Прош. прич. страд. В. мн. одуш.	перефотографированных
142	Прош. прич. страд. В. мн. неодуш.	перефотографированные
143	Прош. прич. страд. Т. мн.	перефотографированными
144	Прош. прич. страд. П. мн.	перефотографированных
145	Прош. прич. страд. кф. мн.	перефотографированы

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

Отчество жен.	*MO_POS_MIDDLENAMEFEMININE*	Васильевна	С
Фамилия	*MO_POS_LASTNAME*	Иванов	П
Географ. название муж.	*MO_POS_GEOGRAPHYNAME MASCULINE*	Орел	С
Географ. название жен.	*MO_POS_GEOGRAPHYNAMEFEMININE*	Обь	С
Географ. название сред.	*MO_POS_GEOGRAPHYNAMENEUTER*	Внуково	С
Географ. название множ.	*MO_POS_GEOGRAPHYNAMEPLURAL*	Карпаты	С
Глагол переходный несов. вида	*MO_POS_VERBIMPERFECT*	смотреть	Г
Глагол непереходный несов. вида	*MO_POS_VERBINTRANSITIVEIMPERFECT*	чихать	Г
Глагол переходный совершенного вида	*MO_POS_VERBPERFECT*	посмотреть	Г
Глагол непереходный совершенного вида	*MO_POS_VERBINTRANSITIVEPERFECT*	чихнуть	Г
Глагол переходный двувидовой	*MO_POS_VERBTWOWAY*	эмитировать	Г
Глагол непереходный двувидовой	*MO_POS_VERBINTRANSITIVE TWOWAY*	эмитрировать	Г
Прил.	*MO_POS_ADJECTIVE1*	фонетический	П
Прил.	*MO_POS_ADJECTIVE2*	фонетический	П
Сущ. неодуш. муж.	*MO_POS_NOUNINANIMATEMASCULINE*	трактор	С
Сущ. одуш. муж.	*MO_POS_NOUNANIMATEMASCULINE*	зяблик	С
Сущ. одуш.-неодуш. муж.	*MO_POS_NOUNANIMATEINANIMATE MASCULINE*	кодировщик	С
Сущ. неодуш. жен.	*MO_POS_NOUNINANIMATEFEMININE*	весна	С
Сущ. одуш. жен.	*MO_POS_NOUNANIMATEFEMININE*	птичка	С
Сущ. одуш.-неодуш. жен.	*MO_POS_NOUNANIMATEINANIMATE FEMININE*	матрешка	С
Сущ. неодуш. сред.	*MO_POS_NOUNINANIMATENEUTER*	озеро	С
Сущ. одуш. сред.	*MO_POS_NOUNANIMATENEUTER*	дитя	С
Сущ. одуш.-неодуш. сред.	*MO_POS_NOUNANIMATEINANIMATE NEUTER*	детище	С
Сущ. неодуш. множ.	*MO_POS_NOUNINANIMATEPLURAL*	ножницы	С
Наименование муж. рода	*MO_POS_APPELATION_MASCULINE*	Гарант-Парк-интернет	С
Наименование жен. рода	*MO_POS_APPELATION_FEMININE*	Формоза	С
Наименование сред. рода	*MO_POS_APPELATION_NEUTER*	Алеко	С
Наименование множ. числа	*MO_POS_APPELATION_PLURAL*	Инфо-Технологии	С