СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Задание | 2 |
| Выбор средств для решения задания | 3 |
| Обоснование выбора средств решения задачи | 4 |
| Алгоритм | 5 |
| Примеры | 8 |
| Код | 10 |
| ПРИЛОЖЕНИЕ 1 | 12 |
| ПРИЛОЖЕНИЕ 2 | 13 |

Задание

Произвести морфологический разбор слов в предложении, указав для каждого слова часть речи (в скобках после соответствующего слова).

Входные данные: предложение на русском языке.

Выходные данные: исходное предложение, в котором после каждого слова в скобках указана часть речи.

Выбор средств для решения задачи

Для решения задачи был выбран язык высокого уровня Python(см. ПРИЛОЖЕНИЕ 1) со средой разработки IDLE и использованием библиотеки pymorphy(см. ПРИЛОЖЕНИЕ 1).

Обоснование выбора средств решения задачи

Языком программирования был выбран Python (см. ПРИЛОЖЕНИЕ 1), потому что этот язык ориентирован на повышение производительности разработчика и читаемости кода. Это очень мощный язык, у него много «фанатов», что предоставляет возможность для консультирования и быстрой помощи в случае возникновения трудностей.

Библиотекой была выбрана pymorphy (см. ПРИЛОЖЕНИЕ 1), так как она очень легка в использовании, имеет множество подключаемых словарей, поддерживает разбор слов с дефисами и со сложными префиксами.

Алгоритм

Все прототипы функций взяты с сайта <http://pythonhosted.org/pymorphy> (<http://pythonhosted.org/pymorphy/ref/morph.html> ) и <http://docs.python.org/2.7> .

Вся справочная информация взята с сайта <http://pythonhosted.org/pymorphy> , <http://rd.feb-web.ru/pollex-12-1.html> и <http://habrahabr.ru/post/49421/> .

1. Считать входные данные (предложение)

Считать входные данные лучше всего либо при вводе с консоли, либо из файла. Удобнее сделать считывание с консоли, так как работа с файлом занимает дополнительное время у пользователя и ресурсы у системы, а при необходимости сохранения выходных данных это можно сделать вручную, скопировав с консоли.

Прототип функции, считывающей данные с консоли:

**raw\_input**([prompt])

1. Разбиение входного текста на токены

Для морфологического анализа необходимо подать в анализатор *слово*.

Так как предложение чаще всего состоит из *нескольких* слов, *знаков* *пунктуации* и *чисел*, необходимо выделить слова.

Прототип функции, разбивающей текст на токены(слова, знаки препинания):

tokenizers.**extract\_tokens**(text)

Пример:

|  |
| --- |
| from pymorphy import get\_morph # портируем модули  from pymorphy.contrib import tokenizers  morph = get\_morph('C:/Python27/ru.sqlite-json/') # создаем объект класса pymorph  text = (raw\_input("Входные данные(): ")).decode('cp1251')  print "Выходные данные():",  for token in tokenizers.extract\_tokens(text): # выделяем токены  print token |

1. Из выделенных токенов находим слова

Токены, найденные на втором шаге алгоритма, содержат не только слова, но еще и знаки пунктуации и пробелы, что не позволяет каждый токен подавать в анализатор. Будем считать, что токен содержит слово, если он состоит только из одних букв, не содержит ни пробелов, ни знаков препинания (кроме дефисов), ни цифр.

Приведем прототип функции, которая проверяет символы строки (строка – это тип данных) на содержание букв. Если все символы являются буквами, функция возвращает *True*(значение булево типа), иначе *False*(значение булево типа).

Прототип этой функции:

s.**isalpha**(), где *s* – строка.

Пример:

|  |
| --- |
| s = u‘дерево’ # строка  if s.isalpha() == True: # проверка символов строки  print str(‘Строка {0} состоит только из букв’).format(s.encode(‘cp1251’))  else:  print str(‘Строка {0} состоит не только из букв(букв может не быть)’).format(s.encode(‘cp1251’)) |

1. Морфологический анализ слова, нахождение части речи

После того, как мы выделили токены, которые, предположительно, содержат слова (смысловые слова), мы подаем их на вход анализатора. Функция, анализирующая слово, возвращает грамматическую информацию о нем и его нормальную форму. Работает функция по словарю, для неизвестных слов работает предсказатель, если возможных форм несколько - возвращает несколько форм.

Прототип этой функции:

**get\_graminfo**(word, standard=False, predict=True, \*\*kwargs)

Есть два ограничения на работу этой функции:

* символы слова должны находиться в формате *Unicode*
* символы слова должны быть *в верхнем регистре* (заглавными)

Слово в *Юникоде* сразу после ввода. Прототип ф-ции, возводящий буквы в верхний регистр: s.**upper**(), где *s* – строка.

Пример:

|  |
| --- |
| from pymorphy import get\_morph # портируем модули  from pymorphy.contrib import tokenizers  morph = get\_morph('C:/Python27/ru.sqlite-json/') # создаем объект класса pymorph  text = (raw\_input("Входные данные(): ")).decode('cp1251')  print "Выходные данные():",  for token in tokenizers.extract\_tokens(text): # выделяем токены  if word.isalpha() == True: # отсеиваем пунктуацию  info = morph.get\_graminfo(word.upper()) # анализ |

1. Выводим токены

Если после проверки токена он состоит только из букв, мы подаем его на анализатор, выводим его, после чего к концу в скобках добавляем часть речи (в верхнем регистре и сокращенном виде, чтобы они выделялись и бросались в глаза), которую возвратим анализатор.

Если после проверки токена он не проходит проверку на то, что он состоит из одних букв, и он не является пробелом, мы просто выводим его.

Прототип функции вывода текста:

**print** expression

1. Предложение вывести расшифровку частей речи

Как говорилось на пятом шаге алгоритма, части речи в выходных данных приписываются в скобках к словам в сокращенном виде (см. Примеры).

Чтобы посмотреть полное название части речи, если оно не понятно из сокращения, после текста с выходными данными предлагается вывести справку с расшифровкой частей речи. Сокращенный вид и расшифровку можно увидеть в ПРИЛОЖЕНИЕ 2.

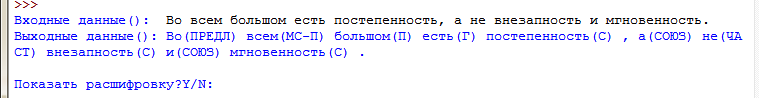
Примеры

Пример №1

Обычное предложение, ничего особенного.

Входные данные: Во всем большом есть постепенность, а не внезапность и мгновенность.

Выходные данные: Во(ПРЕДЛ) всем(МС-П) большом(П) есть(Г) постепенность(С) , а(СОЮЗ) не(ЧАСТ) внезапность(С) и(СОЮЗ) мгновенность(С).

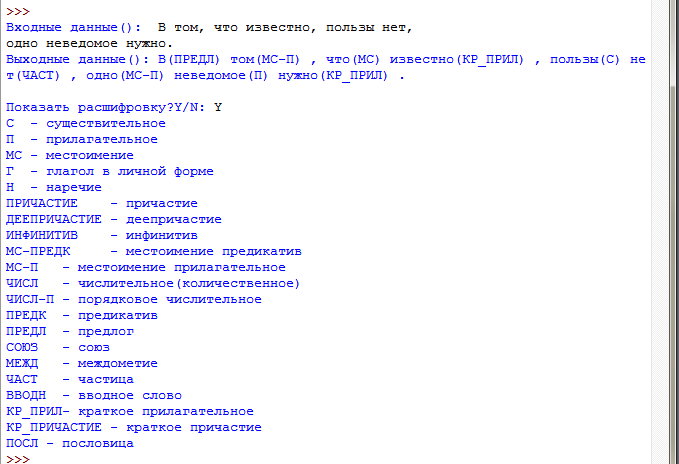


Пример №2.

Предложение посложней, чем предложение из первого примера.

Входные данные: В том, что известно, пользы нет, одно неведомое нужно.

Выходные данные: В(ПРЕДЛ) том(МС-П) , что(МС) известно(КР\_ПРИЛ) , пользы(С) нет(ЧАСТ) , одно(МС-П) неведомое(П) нужно(КР\_ПРИЛ) .

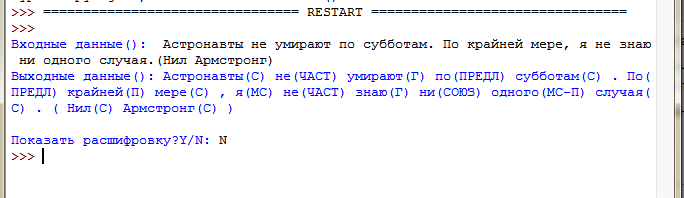


Пример №3.

Иллюстрация реакции программы на фамилию и имя.

Входные данные: Астронавты не умирают по субботам. По крайней мере, я не знаю ни одного случая. (Нил Армстронг)

Выходные данные: Астронавты(С) не(ЧАСТ) умирают(Г) по(ПРЕДЛ) субботам(С) . По(ПРЕДЛ) крайней(П) мере(С) , я(МС) не(ЧАСТ) знаю(Г) ни(СОЮЗ) одного(МС-П) случая(С) . ( Нил(С) Армстронг(С) )

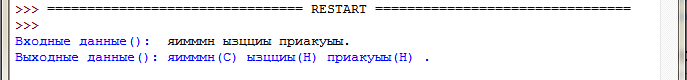


Пример №4.

Реакция программы на ввод несуществующих слов.

Входные данные: яимммн ызцциы приакуыы.

Выходные данные: яимммн(С) ызцциы(Н) приакуыы(Н) .



Код

|  |
| --- |
| from pymorphy import get\_morph # портируем модули  from pymorphy.contrib import tokenizers  morph = get\_morph('C:/Python27/ru.sqlite-json/') # создаем объект класса pymorph  text = (raw\_input("Входные данные(): ")).decode('cp1251')  print "Выходные данные():",  for word in tokenizers.extract\_tokens(text): # выделяем токены  if word.isalpha() == True: # отсеиваем пунктуацию  info = morph.get\_graminfo(word.upper()) # анализ  print str('{0}({1})').format(word.encode('cp1251'), info[0]['class'].encode('cp1251')),  elif word.isspace() == False:  print word,  while True:  q = (raw\_input("\n\nПоказать расшифровку?Y/N: "))  if q == 'Y':  print 'С - существительное'  print 'П - прилагательное'  print 'МС - местоимение'  print 'Г - глагол в личной форме'  print 'Н - наречие'  print 'ПРИЧАСТИЕ - причастие'  print 'ДЕЕПРИЧАСТИЕ - деепричастие'  print 'ИНФИНИТИВ - инфинитив'  print 'МС-ПРЕДК - местоимение предикатив'  print 'МС-П - местоимение прилагательное'  print 'ЧИСЛ - числительное(количественное)'  print 'ЧИСЛ-П - порядковое числительное'  print 'ПРЕДК - предикатив'  print 'ПРЕДЛ - предлог'  print 'СОЮЗ - союз'  print 'МЕЖД - междометие'  print 'ЧАСТ - частица'  print 'ВВОДН - вводное слово'  print 'КР\_ПРИЛ- краткое прилагательное'  print 'КР\_ПРИЧАСТИЕ - краткое причастие'  print 'ПОСЛ - пословица'  break  elif q != 'N':  print 'Некорректный ввод',  else:  break |

ПРИЛОЖЕНИЕ 1

Коротко о выбранных средствах:

**Python** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *python* — [питон](http://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D1%82%D0%BE%D0%BD); в русском языке распространено название *пито́н*)—[высокоуровневый язык программирования](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. [Синтаксис](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ядра Python минималистичен. В то же время [стандартная библиотека](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_Python) включает большой объём полезных функций.

Python поддерживает несколько [парадигм программирования](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в том числе [структурное](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [объектно-ориентированное](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [функциональное](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5),[императивное](http://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [аспектно-ориентированное](http://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%BF%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Основные архитектурные черты — [динамическая типизация](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](http://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0), полная [интроспекция](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), механизм [обработки исключений](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка [многопоточных вычислений](http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и удобные высокоуровневые [структуры данных](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Код в Питоне организовывается в функции и [классы](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые могут объединяться в [модули](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (которые в свою очередь могут быть объединены в пакеты).

(Википедия, <http://ru.wikipedia.org/wiki/Python> )

**pymorphy** - библиотека для морфологического анализа на Python, распространяется по лицензии MIT.

За основу были взяты наработки с сайта aot.ru. Словари (LGPL) для русского и английского, а также идеи - оттуда.

На aot.ru описаны и конкретные алгоритмы реализации, но в терминах теории автоматов. Реализация в pymorphy независимая, не использует конечные автоматы, данные хранятся в key-value хранилище (поддерживаются разные варианты), все алгоритмы переписаны с учетом этого факта.

В pymorphy также есть некоторые возможности, отсутствующие в оригинальной реализации, например:

* поддерживается разбор слов с дефисами, разбор слов со сложными префиксами
* реализовано склонение слов, постановка слов во множественное число

([http://pythonhosted.org/pymorphy/index.html#](http://pythonhosted.org/pymorphy/index.html) )

**IDLE** — это [интегрированная среда разработки](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) на языке [Python](http://ru.wikipedia.org/wiki/Python), созданная с помощью библиотеки [Tkinter](http://ru.wikipedia.org/wiki/Tkinter). Поставляется вместе с Python.

(Википедия, <http://ru.wikipedia.org/wiki/IDLE> )

ПРИЛОЖЕНИЕ 2

Сокращенный вид и расшифровка частей речи:

|  |  |
| --- | --- |
| **Сокращение** | **Расшифровка** |
| С | существительное |
| П | прилагательное |
| МС | местоимение |
| Г | глагол в личной форме |
| Н | наречие |
| ПРИЧАСТИЕ | причастие |
| ДЕЕПРИЧАСТИЕ | деепричастие |
| ИНФИНИТИВ | инфинитив |
| МС-ПРЕДК | местоимение предикатив |
| МС-П | местоимение прилагательное |
| ЧИСЛ | числительное(количественное) |
| ЧИСЛ-П | порядковое числительное |
| ПРЕДК | предикатив |
| ПРЕДЛ | предлог |
| СОЮЗ | союз |
| МЕЖД | междометие |
| ЧАСТ | частица |
| ВВОДН | вводное слово |
| КР\_ПРИЛ | краткое прилагательное |
| КР\_ПРИЧАСТИЕ | краткое причастие |
| ПОСЛ | пословица |