Ruby: строки

Информатика
10-11 классы

10 ноября 2011 г.

Строки

- Строки — наиболее популярный тип в любом языке программирования.
- При вводе / выводе информации любой тип данных преобразуется в строку.
- Имя, фамилия, условие задачи и даже программа — всё это строки.
- Строки очень похожи на массивы. Строка суть набор букв, то есть, массив.
- Поэтому работа со строками столь же проста в ruby, как и с массивами.

История о возможностях ruby

- Студенты четвёртого курса МЭТТ ГАИ поступили на подготовительные курсы в МГИУ. Там им начали преподавать основы программирования на Ruby.
- И одна из заданных им задач была: “Дано число, необходимо поменять порядок цифр на обратный”.
- Задача сложная, но наши студенты об этом не знали и решили её преобразованием к строке: $\mathrm{given.to\_s.reverse}$.
- Преподаватели были поражены и впредь запретили им использовать преобразования к строке в своих программах.
- И всё потому, что это сильно упрощало решение и давало студентам огромное преимущество перед остальными слушателями курсов. (ВикиУчебник)

Ruby — это просто

Создание строк

- Простейший способ задания строки — через кавычки ".
- Можно преобразовать число к строке с помощью to_s.
- В строках можно выводить значение переменных с помощью конструкции *var*

Listing 1: Способы создания строки

```
name = "Ivan Ivanov"
age  = 56.to_s
var  = 35
puts "var = #{var}"
```

Конкатенация (сложение строк)

- Строки можно складывать. Следующая программа выведет на экран строку “Help us Obi Wan Kenobi”:

Listing 2: Конкатенация строк

```
help = "Help us"
puts help + " Obi Wan Kenobi"
```

- Строки можно умножать на целые числа. Пример ниже выведет строку “ахахахах”:

Listing 3: Умножение строк

```
puts "ax"*4
```

Методы работы со строками

- s = “Во дворе - дрова, а в дровах - трава!”.

Метод	Описание (результат)
s.size	количество символов (33)
s[3]	четвёртый (с учётом нулевого) символ (д)
s[−1]	последний символ (в обратную сторону) (!)
s[3..7]	символы с 3-го по 7-й (дворе)
s[3..7].reverse	перевернуть (еровд)
s.sub(“трава”, “мясо”)	заменить одно слово “трава” на слово “мясо”
s.gsub(“а”, “о”)	заменить все “а” на “о”
s.split(“слово”)	разделить строку по слову (вернёт массив подстрок)

Подсчёт количества слов

- Решим простую задачу о подсчёте количества слов в строке.
- Известно, что слова отделяются пробелами.
- Поэтому разобьём строку по пробелам и подсчитаем количество получившихся *подстрок*.

Listing 4: Слова

```
s = "All your bases are belong to us!"
substrings_array = s.split(" ")
puts substrings_array.size
```

Жи-ши пиши с И

- Напишем программу, исправляющую некоторые ошибки в написании.
- Правила будут следующие: “жи–ши пиши с **и**, ча-ща пиши с **а**”.
- Для замены воспользуемся методом gsub.

Listing 5: Правила

```
s = "Начяльника, моя жыть хочет, дай денег на чяй!"
correct_s = s.gsub("жы", "жи").gsub("шы", "ши")
                .gsub("чя", "ча").gsub("щя", "ща")
puts correct_s
```

Метод each

- Для прохода по массивам / строкам помимо уже изученных методов есть ещё несколько весьма полезных.
- Метод each позволяет пройтись по каждому элементу массива и выполнить какое–либо действие.
- По принципу он очень похож на методы map, inject и др.
- Посчитаем длину каждого слова в строке.
- Для этого разобьём строку на слова (в виде массива), а затем пройдёмся по массиву и посчитаем длину каждого слова.

Метод each

Listing 6: Метод each — краткая запись

```
s = "This is SPARTA!"
s_arr = s.split(" ")
s_arr.each { |elem| puts elem.size }
```

Listing 7: Метод each — полная запись

```
s = "This is SPARTA!"
s_arr = s.split(" ")
s_arr.each do |elem|
  puts elem.size
end
```

Ещё несколько полезных методов

- s.trim — убирает лишние пробелы в начале и в конце строки.
- s.empty? — проверяет, пуста ли строка.
- s.include?(other_s) — содержит ли строка подстроку other_s.
- Продвинутые методы:
- s.each do |char| — проходит по всем символам строки.
- s.each_line do |line| — проходит по всем строкам текста.
- s[1].chr — номер ASCII–символа.
- 97.ord — перевод из ASCII–кода в символ.
- \n — символ перевода строки (“ \r\n” в Windows).

Задания

- Напишите программу, выводящую на экран среднюю длину слова в заданной строке (ответ представить в виде целой части получившегося числа).
- Напишите программу, выводящую на экран слова, чья длина превышает среднюю. В конце программы укажите число таких слов.
- (Повышенной сложности) *Палиндромом* называют слово (или буквосочетание), одинаково читающееся в обоих направлениях: топот, *А роза упала на лапу Азора* (Фет). Задача: вывести на экран все палиндромы–слова, встречающиеся в строке.
- (МегаПовышенной сложности) Вывести на экран все палиндромы (включая словосочетания), встречающиеся в строке.

References

- Все презентации доступны на http://school.smirik.ru!
- Вопросы, предложения, д/з: smirik@gmail.com
- Благодарности: ВикиУчебник, Википедия, Гугол, мозг.