Барский Евгений Яковлевич
Тренер команды Московского физико-технического института (МФТИ) по программированию.

Ruby - язык быстрого программирования

Человек создан для творчества, и я всегда знал, что люблю творить. Увы, я обделён талантом художника или музыканта. Зато умею писать программы.

Я хочу, чтобы компьютер был моим слугой, а не господином, поэтому я должен уметь быстро и эффективно объяснить ему, что делать.

Юкихиро Мацумото

Ру́би (с англ. Ruby – «Рубин») – современный интерпретируемый язык программирования высокого уровня, сочетающий в себе полный объектно-ориентированный механизм в стиле Smalltalk, регулярные выражения не менее удобные, чем в Perl, генерацию функций «налету» и передачу таких исполняемых блоков в качестве параметров, свойственную функциональному программированию. Конечно, для большинства языков есть соответствующие библиотеки, однако это сказывается на читабельности программ; средство, встроенное напрямую в синтаксис языка, всегда воспринимается проще и понятнее.

Руби	
Семантика:	мультипарадигменный
Тип исполнения:	интерпретатор
Появился в:	1995 г.
Автор:	Юкихиро Мацумото
Типизация данных:	строгая, динамическая
Основные реализации:	Ruby, JRuby
Диалекты:	отсутствуют
Создан под влиянием:	Perl, Эйфель, Smalltalk, Ада, Dylan, Python, CLU, LISP, C++
Оказал влияние на:	Groovy, Amber

Язык сделан в соответствии с принципом наименьшей неожиданности (principle of least surprise). Это значит, что программа должна по возможности делать ровно то, что ожидает от неё программист, а интерфейсы, создаваемые программистом, должны сами получаться удобными и т.д. На практике попытка внедрить этот принцип приводит к идее, что код программы должен по возможности походить на человеческую речь или хотя бы на математическую. Это значит что для обработки какой-то информации мы должны сформулировать, что мы хотим с ней сделать, перевести на английский язык глаголы и прописать их через точку. Поясним на примере.

Задача. Частоты слов: файл "`a.txt`" содержит слова. Все символы, кроме русских и латинских букв, считайте разделительными символами. Выведите 10 наиболее часто встречаемых слов длины более 6 символов с указанием их частот в порядке убывания частоты.

```
File.open("a.txt").read.
scan(/[A-Za-z]+/).
select {|w| w.size>6}.
    inject(Hash.new(0)) {|f,word|
    f[word]+=1
    f
}.to_a.sort_by {|x| x[1]}.reverse[0..9].each {|pair|
     printf("%10s %5d\n", pair[0], pair[1])
}
```

Это достаточно сложный код. Человеку, не знающему языка, сложно будет понять все детали. Но посмотрите, как легко он читается.

Начало кода

Работать будем с файлом	`File`
Откроем a.txt	`open("a.txt")`

Продолжение кода на следующей странице

Продолжение кода

Прочтём содержимое	`read`
Прочтём из содержимого латинские слова	`scan(/[A-Za-z]+/)`
Выберем те слова **w**, длина которых больше шести	`select {|w| w.size>6}`
Поместим массив в новый хеш, у которого значения по умолчанию равны 0	`inject(Hash.new(0))`

Имеет текущий хэш частот **f** и новое считанное слово **word**. На каждом шаге увеличиваем частоту слова **word** на 1, результатом будет обновлённый хэш **f**.	`{|f,word|`
	`  f[word]+=1`
	`  f`
	`}`

Преобразуем хеш в массив пар [слово, частота]	`to_a`
Отсортируем по второму элементу	`sort_by {|x| x[1]}`
Перевернём массив и возьмём первые 10	`reverse[0..9]`

Каждую пару **p** форматировано выведем	`each {|p|`
	`printf("%10s %5d\n",p[0],p[1])`
	`}`

Этот код можно несколько оптимизировать:

```
File.open("a.txt").read.scan(/[A-Za-z]{6,}/).
    inject(Hash.new(0)){|f,word|
      f[word]+=1
      f
}.to_a.sort_by {|x| -x[1]}[0..9].each { |pair|
      printf("%10s %5d\n", pair[0], pair[1])
}
```