Ruby 初級者向けレッスン第 17 回

okkez @ Ruby 関西, チカホリ@小波ゼミ

2008 年 02 月 16 日

今回の内容

- 少しだけ前回の復習
- イテレータの種類の紹介
- るりま
- 演習 (gropu work)
- まとめ

今回のゴール

- イテレータの種類を知る
- そのうちいくつかを使用してみる

ざっくり前回の復習

- 2 から 1000 までの素数を画面に出力
- 素朴な実装から徐々に最適化、高速化
- 複数の実装を Benchmark して比較
- 約 40 倍高速化した
- 演習で unix の cal コマンド作成

ベンチマークの結果 (おまけ)

100 回実行。

```
user     system      total        real
```

```
prime0?:   75.290000  13.260000  88.550000 (141.615915)
prime1?:    7.810000   0.920000   8.730000 ( 14.184665)
prime2?:    2.750000   0.340000   3.090000 (  5.282121)
prime3?:    2.370000   0.310000   2.680000 (  4.133397)
prime4?:    1.860000   0.200000   2.060000 (  3.269983)
prime5?:    1.700000   0.220000   1.920000 (  3.036675)
```

ざっと 40 倍高速化してる。実装の詳細は前回の資料を参照してください。

イテレータの種類

- 要素アクセス型
- アクセスユーティリティ型
- 範囲型
- 登録型
- パラメータ型
- 仕事分割型
- コンテキスト型

要素アクセス型

典型的な例は Array#each、Hash#each、String#each_byte など。

コンテナオブジェクトが、すでに自分の保持しているオブジェクトに対してのアクセス手段を提供する。最も原始的なイテレータの用法でもある。引数を破壊的に変更すると要素自体が変更されるようになることが多い。

アクセスユーティリティ型

典型的な例は Array#each_index、String#each、Integer#upto など。

オブジェクトを次々にブロックにあたえる点は要素アクセス型と同じだが、そのオブジェクトが「作られた」ものであることが違う点。例えば、 Array#each_index はブロックの引数は Array に関係してはいるけれども、持っているわけではない。それゆえ、要素アクセス型とは違って、引数のオブジェクトを破壊的に変更しても副作用がおきないことが多い。このタイプのメソッドはたいていユーザレベルでも定義できるのだが、ユーザの簡便のために提供されている、すなわちユーティリティである。

範囲型

典型的な例は IO#open、timeout など。

イテレータブロックの実行中だけ特定の環境を設定する。必然的にこのタイプのイテレータは「一回だけくりかえされるイテレータ」であることが多い。開始と終了があるものにはほぼまちがいなくこのイテレータが適用できる。特に終端が必須であるものには、ensure とくみあわせて確実に終端が行われるイテレータを提供するとよい。

また、Execute Around Method とも言うらしい。

実装例

よく使う例として、win32ole で Excel を操作する場合に例外処理を確実に行う場合の実装例。

```
# excel.rb
require 'win32ole'

module Excel
  def self.run_during(visible = true, display_alerts = false, &block)
    begin
      excel = new(visible, display_alerts)
      block.call(excel)
    ensure
      excel.Quit
    end
  end
end

# sample.rb
require 'win32ole'
require 'excel'

Excel.run_during(false, false) do |excel|
  # Excel を操作する処理
end
```

登録型

典型的な例は signal、at_exit、Gtk::Widget#signal_connect など。

いわゆるコールバックルーチンの登録である。

パラメータ型

典型的な例は Enumerable#collect、Array#delete_if など。

オブジェクトをパラメータ化したい時は引数を渡す。一方、コードをパラメータ化したい時はイテレータを使ってブロックを渡す。テンプレートメソッドとかストラテジーと呼ばれるものがこの範疇である。このタイプはブロックの返り値に意味があることが多い。

仕事分割型

実例は IO#each、SMTP#sendmail など。

結果全部を一気に返すとでかすぎて危険な時に、文字列 (とか) を少しづつ渡すタイプ。IO#each はどっちかというと行に分割するほうが目的だと思うが、そういう目的にも (確実ではないが) 使えるので挙げてみた。このタイプの特徴として、ブロックの引数を再現できないことが多い。

コンテキスト型

典型的な例は module_eval と instance_eval.

このタイプはブロックを eval することが目的といっていい。他の例はあまり見られないのだが、おれは結構使ってる。使いたくなるのは、例えば「ちょっとした環境が欲しい」「ここだけで通用するメソッドを定義したい」「使い捨てにしたいからいちいちクラスは作りたくない」なんて時に有効である。簡単な使用例をあげよう。

```
def mkenv( &block )
  Object.new.instance_eval(&block)
end

mkenv {
  def put( msg )
    $stderr.puts msg
  end
  put 'test, test...'
  put 'tadaima maikuno tesuto tyuu.'
}
```

るりま

- 先日、無事初版 (α 版) をリリースしました
- 組込みのクラスやメソッドにはほぼ対応しています
- 1.8 の HEAD に合わせてあります
- 1.9 対応は今後の課題です

How to get it

- 以下の URL からダウンロードしてください
- http://doc.loveruby.net/wiki/ReleasedProducts.html
- Windows ユーザーの人は chm 版が便利です

How to use it

- chm は検索ができるので検索してください
- dynamic 版はクラス名、モジュール名やメソッド名で探してください
- 詳しい説明は以下を参照してください
- http://doc.loveruby.net/wiki/ReleasePackageHowTo.html

How to join the project

- ML に参加して
- 青木さんにメールして svn のアカウント発行してもらう
- 詳細は以下を参照してください
- http://doc.loveruby.net/wiki/FrontPage.html

演習問題

以下の問題では for, while などは使用禁止です。

九九の表

九九の表を作成しよう。
以下のような出力を得られるプログラムを作成してください。

```
|  1  2  3  4  5  6  7  8  9
--+--------------------------
 1|  1  2  3  4  5  6  7  8  9
 2|  2  4  6  8 10 12 14 16 18
 3|  3  6  9 12 15 18 21 24 27
 4|  4  8 12 16 20 24 28 32 36
 5|  5 10 15 20 25 30 35 40 45
 6|  6 12 18 24 30 36 42 48 54
 7|  7 14 21 28 35 42 49 56 63
 8|  8 16 24 32 40 48 56 64 72
 9|  9 18 27 36 45 54 63 72 81
```

100 マス計算

100 マス計算の問題と解答を作成しよう。
以下のような出力を得られるプログラムを作成してください。

```
+|  8  2  1  4  5  0  3  7  6  9

--+------------------------------
 8|
 2|
 0|
 9|
 4|
 7|
 6|
 3|
 5|
 1|


 +|  8  2  1  4  5  0  3  7  6  9
--+------------------------------
 8| 16 10  9 12 13  8 11 15 14 17
 2| 10  4  3  6  7  2  5  9  8 11
 0|  8  2  1  4  5  0  3  7  6  9
 9| 17 11 10 13 14  9 12 16 15 18
 4| 12  6  5  8  9  4  7 11 10 13
 7| 15  9  8 11 12  7 10 14 13 16
 6| 14  8  7 10 11  6  9 13 12 15
 3| 11  5  4  7  8  3  6 10  9 12
 5| 13  7  6  9 10  5  8 12 11 14
 1|  9  3  2  5  6  1  4  8  7 10
```

- 計算の種類は左上の隅に出力する
- 計算の種類は四則 (加減乗除) とする (+, -, *, /)
- 除算の場合はゼロ割の可能性があるので考慮すること
- 出力される結果は毎回異なる (ランダム) ものとする

色見本作成

rgb.txt を使用して HTML の色見本を作成してください。自分の PC に rgb.txt が入っている人はそれを使用してください。
なお、valid な html である必要はありません。
以下に rgb.txt の最初の数行を示します。

```
! $Xorg: rgb.txt,v 1.3 2000/08/17 19:54:00 cpqbld Exp $
255 250 250 snow
248 248 255 ghost white
248 248 255 GhostWhite
245 245 245 white smoke
```

ヒント

リファレンスマニュアルで以下のクラスやメソッドについて調べてみてください。

- Range
- Array#map
- Array#collect
- Array#each
- Array#each_with_index
- Array#join
- Array#sort_by
- Kernel.#rand
- Kernel.#printf
- Kernel.#sprintf
- Kernel.#__send__
- String#%
- String#split

参考文献

極めるイテレータ：
http://i.loveruby.net/ja/ruby/iterator.html

Ruby 初級者向けレッスン 第 16 回：
http://docs.google.com/Present?docid=ddv4xrfg_5hdf6jrfn

Ruby による Win32OLE プログラミング - Excel プログラミング（初級編）
http://www.morijp.com/masarl/homepage3.nifty.com/masarl/article/ruby-win32ole/excel-1.html