Ruby 初級者向けレッスン 第 12 回

okkez @ Ruby 関西, サカイ@小波ゼミ

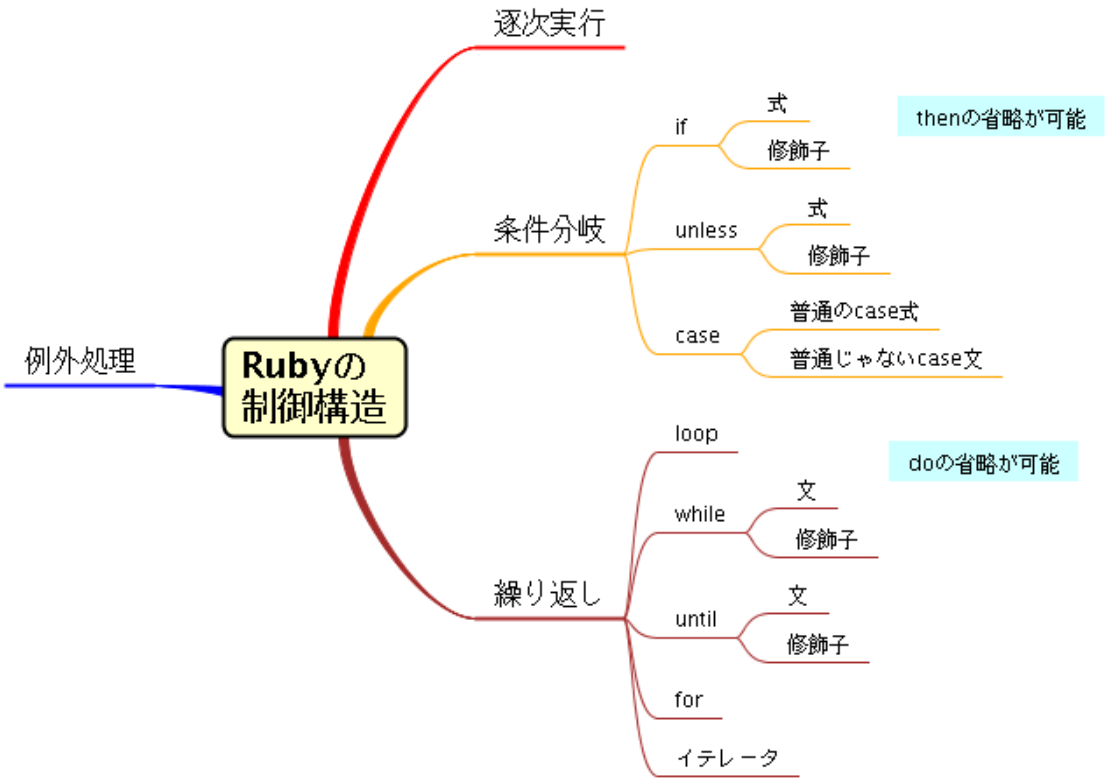
2007 年 04 月 28 日

制御構造

Ruby の制御構造には以下のものがあります。

- 逐次処理
- 条件判断
- 繰り返し
- 例外処理

今回は、上から三つまでを取り扱います。



逐次処理

プログラムを書かれた通りに、先頭から順に実行する

```
n = 1
puts  "#{n}番目に実行する"
n += 1
puts  "#{n}番目に実行する"
n += 1
puts  "#{n}番目に実行する"
```

条件判断

ある条件が成り立つ場合は○○を、そうでない場合は××を実行する

```
if 条件 then
  条件が成り立ったときに実行したい処理
else
  条件が成り立たなかったときに実行したい処理
end
```

```
param = 10

if param > 10
  puts 'param > 10'
else
  puts 'param <= 10'
end
```

このように then を省略することができます。また以下のように、 else 節を省略することができます。

```
# 引数がいくつかあるコマンドを作成する場合
if ARGV.size == 0
  raise ArgumentError, '引数の数が足りません'
end

puts "引数は #{ARGV.size} 個でした"
```

その他の条件判断

- unless
  - if の逆

- case
  - 条件分岐がたくさんある場合に用いる
- 三項演算子

引数が一つだけのコマンドを作成する場合

```
unless ARGV.size == 1
  raise ArgumentError
end

puts "引数は #{ARGV.size} 個でした"
```

単純に条件が多い場合

```
case
when '1' == ARGV.first then puts '1'
when /\A--help\z/ =~ ARGV.first then puts 'Help me!'
when 100 < ARGV.first.to_i then puts '100 より大きい数字です'
else puts '条件にあてはまりません'
end
```

よく使われるパターン

```
case ARGV.first
when /\A-+h/ then puts 'Help me'
when /\A-+v/ then puts 'version 1.0'
else puts '条件にあてはまりません'
end
```

三項演算子の例

```
ARGV.size == 0 ? puts '引数は 0 個' : puts '引数は 0 個ではない'
```

繰り返し

ある条件が成り立つ間、○○を繰り返し実行する

```
while 条件 do
  条件が成り立っている間ずっと実行したい処理
end

while true
  puts 'ずっと実行される'
end
```

このように do を省略することができます。

その他の繰り返し

loop

- do を省略できない

until

- while の逆

for

- 構文糖 (シンタックスシュガー) のようなもの
- ブロック付きのイテレータとは変数のスコープが異なる

イテレータ

- Integer#times
- Integer#step
- Array#each

などなど

便利な修飾子

if/unless, while/until には短くてきれいなコードを書くのに便利な方法があります。まずは例を示します。

```
[1,2,3,4,5].each do |elem|
  puts elem if elem % 2 == 0 #=> 偶数だけ表示
end
```

このように、式の後ろに修飾子として if 式を記述すると、条件が成り立つ場合のみその直前に書かれた式を実行することができます。

ただし、while/until を修飾子として使用するときは注意が必要です。修飾している文が begin/end ブロックの場合は、条件式の値に関係なく、そのブロック内のコードが常に最低 1 回は実行されます。

```
puts 'Hello' while false
begin
  puts 'Goodbye'
end while false
```

条件の書き方いろいろ

最後に、条件の書き方を説明しておきます。
Ruby では、nil, false 以外は全て真として扱われます。例えば、以下のような感じです。

```
if ARGV.first
  puts 'ARGV.first is not nil'
else
  puts 'ARGV.first is nil'
end
```

この場合は、スクリプト実行時にコマンドライン引数を指定しているかどうかで結果が変化します。コマンドライン引数が指定されていない場合は、 ARGV = [] となるので ARGV.first(ARGV[0]) は nil になります。

```
while line = gets
  # line を処理する
end
```

こちらの例は、ファイルの次の行を返す IO#gets は、ファイルの終端に達すると nil を返すということを利用しています。
また、 Ruby では 0 や空文字列 (””) は偽になりません。

```
if 0
  # 常にに実行される
end

if ''
  # 常に実行される
end
```

これらを、条件に使用したい場合は例えば以下のようにします。

```
param = 1
if param == 0
  # 実行されない
end

value = 'hoge'
if value.empty?
  # 実行されない
end
```

まだまだ色々ありますが、今回はここまでです。

演習

99 Bottles of Beer on the Wall (並)

”99 本のビールが壁に...” 遠足などでよく歌われる古典的な童謡の歌詞、”99 Bottles of Beer on the Wall” を出力するプログラムを書いてみましょう。答えは一つではないので、何通りも書いてみましょう。

```
# 出力例
99 Bottles of beer on the wall
99 Bottles of beer
Take one down and pass it around
98 Bottles of beer on the wall

98 Bottles of beer on the wall
98 Bottles of beer
Take one down and pass it around
97 Bottles of beer on the wall

...

3 Bottles of beer on the wall
3 Bottles of beer
Take one down and pass it around
2 Bottles of beer on the wall

2 Bottles of beer on the wall
2 Bottles of beer
Take one down and pass it around
1 Bottle of beer on the wall

1 Bottle of beer on the wall
1 Bottle of beer
Take one down and pass it around
0 Bottles of beer on the wall
```

石取りゲーム (難)

テイルズシリーズでお馴染みの石取りゲームを Ruby で実装してみましょう。互いにミスをしなければ先手必勝です。

ルール

- コンピュータと人間が対戦する

- 人間は先手か後手か選択できる
- 交互に石を取り合う
- 最後の一つを取った方が負け
- 一度に取れる石の個数は、 1 から 3 個

仕様

- 初期の石の個数は、10 から 100 個の間でランダムにする
- 毎ターン、残りの石の個数を表示する
- 勝負がつくと勝ち負けを表示して終了する

参考文献

初めてのプログラミング

http://www.oreilly.co.jp/books/4873112923/

プログラミング Ruby 第 2 版 言語編

http://ssl.ohmsha.co.jp/cgi-bin/menu.cgi?ISBN=4-274-06642-8

たのしい Ruby 第 2 版

http://shop.sbcr.jp/bm_detail.asp?sku=4797336617

今後の情報源

公式 Web サイト

http://www.ruby-lang.org/

リファレンスマニュアル

http://www.ruby-lang.org/ja/man/

日本 Ruby の会

http://jp.rubyist.net/

okkez のサイト (作成中)

http://fullmetal.dip.jp/