

4.4 例：ニュートン法による平方根計算

ここまでで紹介した構文を、もう少し興味深いプログラムを組み立てて、示しましょう。課題は x の平方根を計算する関数

```
def sqrt(x: Double): Double = ...
```

を書くことです。

平方根を計算する一般的な方法は、近似を繰り返すニュートン法です。まず初期推定値 y (たとえば $y=1$) から始めます。次いで現在の推定値 y を、 y と x/y の平均値を取って繰り返し改良します。次の例では 2 を近似する、推定値 y 、商 x/y 、その平均が 3 列に示されています。

```
1           2/1 = 2           1.5
1.5         2/1.5 = 1.3333    1.4167
1.4167      2/1.4167 = 1.4118  1.4142
1.4142      ...
y           x / y           ( y + x / y )/2
```

Scala では、このアルゴリズムを小さな関数群によって実装でき、各関数がアルゴリズムの各要素を表すようにできます。

はじめに、推定値から結果を得ることを繰り返す関数を定義します。

```
def sqrtIter(guess: Double, x: Double): Double =
  if (isGoodEnough(guess, x)) guess
  else sqrtIter(improve(guess, x), x)
```

`sqrtIter` は自分自身を再帰的に呼び出します。命令型プログラムのループは常に、関数型プログラムでは再帰でモデル化できます。

`sqrtIter` の定義には、パラメータ部に続いて戻り値型があることに注意して下さい。このような戻り値型は再帰関数では必須です。非再帰関数では戻り値型はオプションであり、もしそれがなければ、型チェッカーが関数の右辺から計算します。しかし非再帰関数であっても、よりよい文書化のために戻り値型を書いておくことは、しばしばよい考えです。

二つ目のステップとして、`sqrtIter` から呼ばれる 2 つの関数を定義します。推定値を改良する関数 `improve` と、終了テスト `isGoodEnough` です。定義は次ようになります。

```
def improve(guess: Double, x: Double) =
  (guess + x / guess) / 2

def isGoodEnough(guess: Double, x: Double) =
  abs(square(guess) - x) < 0.001
```

最後に、`sqrt` 関数自身を `sqrtIter` の適用として定義します。

```
def sqrt(x: Double) = sqrtIter(1.0, x)
```

演習 4.4.1 `isGoodEnough` の判定は小さな数に対してはあまり正確ではなく、大きな数に対しては終了しないかもしれません (何故でしょう?)。これらの問題のない別の `isGoodEnough` を設計しなさい。

演習 4.4.2 式 `sqrt(4)` の実行をトレースしなさい。

[前ページ](#) [4章](#) [目次](#) [次ページ](#)

名前:	<input type="text"/>
コメント:	<input type="text"/>

投稿