11 円周率 1 万桁の計算

まず、**PowerShell** による円周率を計算するスクリプトを提示しよう。見出しは円周率 1 万桁となっているが、実はスクリプトは 1000 桁限定のコードである。これを 1 万桁、もしくはそれ以上の任意の桁数で計算させるスクリプトを書くのは君たちの仕事である。旅は、自分の足でするものだよ。そのヒントのためにも、多少の説明をしておこう。

[ps script]

```
$ARRAY = 100; $FIGS = 10000000000
$p = @(); 0..($ARRAY+1) | foreach {$p += 0}
$t = @(); 0..($ARRAY+1) | foreach {$t += 0}

function Div($a, $n) {
  foreach ($i in 0..$ARRAY) {
    $u[$i+1] = $u[$i+1] + ($u[$i] % $a) * $FIGS
    $u[$i] = [math]::floor($u[$i] / $a)
  } foreach ($i in 0..($ARRAY+1)) {$t[$i] = $u[$i]}
  foreach ($i in 0..$ARRAY) {
    $t[$i+1] = $t[$i+1] + ($t[$i] % $n) * $FIGS
    $t[$i] = [math]::floor($t[$i] / $n)
  }
}

function pInc {
  for ($i = ($ARRAY+1); $i -gt 0; $i--) {
    $p[$i-1] = $p[$i-1] + [math]::floor(($p[$i] + $t[$i]) / $FIGS)
    $p[$i] = ($p[$i] + $t[$i]) % $FIGS
  }
}

function pDec {
  for ($i = ($ARRAY+1); $i -gt 0; $i--) {
    $p[$i-1] = ($p[$i-1] - 1) +
                    [math]::floor(($FIGS + $p[$i] - $t[$i]) / $FIGS)
    $p[$i] = ($FIGS + $p[$i] - $t[$i]) % $FIGS
  }
}
```

ここまではスクリプトの前半、長い桁数をもつ数の割り算、足し算、引き算を計算する関数を定義した部分だ。`$p = @();` で円周率を格納する**配列**を準備し、102 個の配列に 0 を入れて初期化してある。`$p[0]` から `$p[101]` までの 102 個であることに注意してほしい。`$t` は臨時計算用の配列だ。いずれの配列にも 10 桁の

数を入れることを想定しているので、全部で 1020 桁分の数を格納できる。余分な配列が 2 つ必要なのは、足し算の繰り上がりと割り算の余残りを格納するためである。

関数 function Div($a, $n) は、101 個の配列を順に割っていくルーティンである。$u と$t の 2 種類の配列を用いて割り算をしているのは、$u は次の項でさらに 5^2 もしくは 239^2 で割る必要があるので、$\frac{1}{2k-1}$ の計算が$u に影響しないように、別の配列$t で行う必要があるのだ。割り算には端数が出るので、[math]::floor を用いて整数値に直している。**PowerShell** では、int は四捨五入によって整数値にする関数である。切り捨てなら floor、切り上げなら ceiling を使う。

割り算の仕組みが飲み込めれば、加算の関数 function pInc と減算の関数 function pDec がやっていることは想像できるはずだ。さて、実際の円周率の計算はこれらの関数を使って行われる。以下はスクリプトの後半である。

[ps script]

```
$u = @(); foreach ($i in 0..($ARRAY+1)) {$u += 0}; $u[1] = (16*$FIGS * 5)
foreach ($i in 1..716) {
  if (($i % 2) -eq 0) {Div (5*5) (2*$i-1); pDec}
  else {Div (5*5) (2*$i-1); pInc}
}

$u = @(); foreach ($i in 0..($ARRAY+1)) {$u += 0}; $u[1] = (4*$FIGS * 239)
foreach ($i in 1..211) {
  if (($i % 2) -eq 0) {Div (239*239) (2*$i-1); pInc}
  else {Div (239*239) (2*$i-1); pDec}
}

$q = @(); foreach ($i in 1..$ARRAY) { $q += ('{0:d10}' -f [long]$p[$i]) }
[string]$p[0] + "."; "$q"
```

円周率の計算は主に、$\frac{16}{5^{2k-1}}$ に絡む計算と $\frac{4}{239^{2k-1}}$ に絡む計算に分かれている。そのために、$\frac{16}{5^{2k-1}}$ に対しては 716 回の計算を行い、$\frac{4}{239^{2k-1}}$ に対しては 211 回の計算で済ませている。1000 桁の精度ならこの程度の計算でよい。

計算が終わると、数値は配列に分割格納されているので、表示の際は見やすくなるように整えたい。そのために別の変数$q を用意し、表示に備えている。変数$q には、次々と配列の値が結合されて、結果は 10 桁ごとに区切られて出力されるはずだ。ただし、最後の配列に相当する末尾 10 桁の半分ほどは数値が怪しくなっている。計算誤差が最後の配列に集積されるからである。もし、確実に 1000 桁の円周率が欲しければ、配列をもうひとつ増やしておくのがよい。

このスクリプトは少々長いので、実際に試す場合はファイルに保存しておこう。その際、前半・後半をつな

げてひとつのファイルにしてほしい。それをたとえば、`pivalue.ps1` という名前で保存したら、プロンプトに対して`./pivalue` と打ち込む。ファイル名の先頭に`./`を付けるのは、セキュリティに厳しい **PowerShell** の作法である。

入力ミスがなければ、環境にもよるが数十秒から 1 分程度して 1000 桁分の円周率が表示される。エラーが出たら、ファイルを開いて訂正した後で保存し直したものを実行する。長いスクリプトでは訂正をする確率が高いので、ファイルに保存しておかないと大変なのだ。