tabularcalc

v0.2

Hướng dẫn sử dụng

Christian Tellechea
unbonpetit@gmail.com
Dịch bởi Lê Hữu Điền Khuê
huudienkhue.le@gmail.com

20.4.2009

Tóm tắt

Được trang bị bởi một danh sách các số và một (hoặc nhiều) công thức (hay hàm số) một biến, tabularcalc giúp chúng ta xây dựng các bảng giá trị bằng các cú pháp rất đơn giản. "Bảng giá trị" ở đây được hiểu là bảng có hàng đầu tiên chứa các giá trị của biến số và các hàng còn lại chứa giá trị của các hàm theo biến số, tương ứng với mỗi giá trị của biến số :

x	-4	-2	0	2.25	7
$f(x)=2x-3$	-11	-7	-3	1.5	11
x^2	16	4	0	5.062,5	49
$h(x)=\sqrt{x^2+1}$	4.123,106	2.236,068	1	2.462,214	7.071,068

Một bảng có thể được xây dựng theo chiều ngang hay chiều dọc và có thể thay đổi theo ý muốn (chiều cao của một ô, kiểu hàng, kiểu cột,...). Hơn nữa, chúng ta có thể ẩn đi giá trị của một (hay một số) ô bất kì. Ngoài ra, đối với mỗi ô, bất kì dòng lệnh nào của TEX cũng đều có hiệu lực, do đó gói này còn có nhiều chức năng khác nữa, tuỳ theo ý muốn của người sử dụng.

Mục lục

Tác giả xin gửi lời cảm ơn đến Derek O'Connor, người đã thử nghiệm phiên bản bêta và đóng góp rất nhiều ý kiến có giá trị.

1 Mở đầu

1.1 Giới thiệu

Gói tabularcalc cho phép xây dựng rất dễ dàng các bảng giá trị của hàm số một biến ứng với mỗi giá trị của biến. Các bảng được trình bày dưới môi trường chuẩn của `tabular` và các giá trị được hiển thị dưới dạng thập phân.
tabularcalc cần đến LaTeXvà các gói phụ trợ `fp`, `xstring` và `numprint`.
Gói này được tạo ra hoàn toàn không phải để cạnh tranh với `pgfplotstable`, một gói rất tuyệt vời của Christian Feuersänger. So với tabularcalc, gói này có thể nói là mạnh hơn nhiều, tuy nhiên các câu lệnh của nó khá phức tạp. Còn tabularcalc, mặc dù khá giản dị, nhưng lại rất dễ sử dụng.

Về mặt hiển thị kết quả với các chữ số thập phân, theo tác giả gói `numprint` là tốt nhất, do đó nó cũng được gọi vào theo mặc định. Chúng ta cũng có thể thay đổi gói này bằng một gói khác, hoặc thay đổi cách hiển thị các giá trị theo ý muốn (xem trang 10).

1.2 Về gói `fp`

Về mặt tính toán, với TEX, tabularcalc không thực hiện việc tính trực tiếp giá trị của các biểu thức, chẳng hạn như `2*x*x-5*x+7` khi `x = 2.7`, mà nó sẽ gọi thêm một "máy tính" phụ trợ là `fp`.

Chúng ta có thể sử dụng cú pháp *trung tố* (infix) hoặc *hậu tố* (postfix) để nhập vào các hàm số. Xem thêm tập tin `README` của gói `fp` để hiểu rõ thêm hai khái niệm trên cũng như danh sách đầy đủ các hàm số có thể sử dụng.

Tuy nhiên, khi làm việc với hàm số mũ, gói này có hai lỗi sau đây :

- thêm vào một khoảng trống không như ý muốn;
- để tính a^b, `fp` sử dụng công thức $a^b = \mathrm{e}^{b \ln a}$. Trong trường hợp b là một số nguyên và a là một số âm, chẳng hạn $(-3)^2 = \mathrm{e}^{2\ln(-3)}$, thì rõ ràng nó sẽ không tính được, bởi vì lô-ga-rít của một số âm là không xác định.

Để sửa hai lỗi này, chúng ta cần thêm vào tuỳ chọn `fixFPpow` như sau :

```
1 \usepackage[fixFPpow]{tabularcalc}
```

1.3 Một số điểm mới

Phiên bản trước của tabularcalc không tương thích được với một số gói khác bởi vì tên của một số macro bị trùng. Do vậy, để tránh vấn đề tương tự với phiên bản này, tác giả đã quyết định đổi tên của tất cả các macro bằng cách thêm vào "`tc`" (viết tắt của **tabularcalc**) ở trước tên cũ, như trong bảng sau :

Tên cũ	Tên mới
`\noshowmark`	`\tcnoshowmark`
`\startline`	`\tcatbeginrow`
`\resetcellcode`	`\tcresetcellcode`
`\listsep`	`\tclistsep`
`\printvalue`	`\tcprintvalue`
`\printresult`	`\tcprintresult`
`\sethrule`	`\tcsethrule`
`\resethrule`	`\tcresethrule`
`\setcoltype`	`\tcsetcoltype`
`\resetcoltype`	`\tcresetcoltype`

Các điểm mới khác :

- "máy tính" `pgfmath`, do độ chính xác kém, đã được thay thế bằng `fp`;
- các giá trị của biến có thể được nhập vào một cách tổng quát bằng công thức;
- kể từ bây giờ, chúng ta có thể soạn thảo lệnh của một bảng trong một tập tin riêng, điều này giúp ta sửa đổi dễ dàng các bảng.

1.4 Thuật ngữ

Trong phần tiếp theo của bài viết, để bạn đọc dễ theo dõi, chúng ta quy ước những số màu đỏ là các "giá trị" của biến, màu xanh là "kết quả" (giá trị cần tính), các biểu thức màu nâu là các "nhãn" (biểu thức cần tính) và ô góc trên cùng bên trái của một bảng gọi là "ô (0,0)", chẳng hạn như hai bảng sau đây:

Bảng ngang

ô (0,0)	-5	-1	0	3	10
x	-5	-1	0	3	10
$2x$	-10	-2	0	6	20
$3x$	-15	-3	0	9	30

Bảng dọc

ô (0,0)	x	$2x$	$3x$
-5	-5	-10	-15
-1	-1	-2	-3
0	0	0	0
3	3	6	9
10	10	20	30

2 Chức năng cơ bản

2.1 Bảng ngang

Lệnh \htablecalc cho phép tạo ra một bảng ngang trong đó hàng đầu tiên chứa giá trị của biến và các hàng còn lại chứa các kết quả. Cú pháp:

\htablecalc[⟨*nb*⟩]{⟨*ô (0,0)*⟩}⟨*biến=danh sách*⟩
{⟨*nhãn 1*⟩}{⟨*công thức 1*⟩}
{⟨*nhãn 2*⟩}{⟨*công thức 2*⟩}
...
{⟨*nhãn n*⟩}{⟨*công thức n*⟩}

trong đó:

- ⟨*nb*⟩ là số công thức cần tính (bằng 1 theo mặc định);
- ⟨*ô (0,0)*⟩ là nội dụng của ô (0,0);
- ⟨*biến*⟩ là tên của biến, tên này sẽ được sử dụng trong các ⟨*công thức*⟩.
- ⟨*danh sách*⟩ là danh sách các giá trị của biến, cách nhau bởi dấu phẩy (Chú ý: dấu phẩy trong các số thập phân phải được thay bằng dấu chấm), nếu có hai dấu phẩy liên tiếp nhau thì sẽ có một cột trống được in ra. Chúng ta có thể thay dấu phẩy bằng một kí tự khác, chẳng hạn với "|", ta dùng lệnh \def\tclistsep{|}.
- ⟨*nhãn i*⟩ là nhãn thứ i;
- ⟨*công thức i*⟩ là công thức thứ i cần tính (tương ứng với hàng thứ i).

Dưới đây là một ví dụ về bảng ở trang đầu tiên của tài liệu :

```
\htablecalc[3]{$x$}{x=-4,-2,0,2.25,7}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

x	-4	-2	0	2.25	7
$f(x) = 2x - 3$	-11	-7	-3	1.5	11
x^2	16	4	0	5.062,5	49
$h(x) = \sqrt{x^2 + 1}$	4.123,106	2.236,068	1	2.462,214	7.071,068

Chúng ta thấy rằng bảng này không hoàn toàn giống với bảng ở trang đầu : độ rộng của các cột không bằng nhau và đường kẻ ngang dưới hàng thứ nhất cũng khác nhau. Chúng ta sẽ tìm hiểu về cách thay đổi các đặc điểm này trong phần sau của tài liệu.

2.2 Bảng dọc

Lệnh \vtablecalc cho phép tạo ra một bảng dọc trong đó cột đầu tiên là giá trị của biến và các cột còn lại là các kết quả. Cú pháp :

\vtablecalc[⟨*nb*⟩]{⟨*ô (0,0)*⟩}⟨*biến=danh sách*⟩
{⟨*nhãn 1*⟩}{⟨*công thức 1*⟩}
{⟨*nhãn 2*⟩}{⟨*công thức 2*⟩}
...
{⟨*nhãn n*⟩}{⟨*công thức n*⟩}

trong đó các khái niệm trong cú pháp trên đều được xác định như trong mục 2.1 *Bảng ngang*.

Sau đây là ví dụ về một bảng dọc của hai bảng trước, trong đó ta sử dụng tên biến là y :

```
\vtablecalc[3]{$x$}{y=-4,-2,0,2.25,7}
              {$f(x)=2x-3$}{2*y-3}
              {$x^2$}{y*y}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,y*y+1),6)}
```

x	$f(x) = 2x - 3$	x^2	$h(x) = \sqrt{x^2 + 1}$
-4	-11	16	4.123,106
-2	-7	4	2.236,068
0	-3	0	1
2.25	1.5	5.062,5	2.462,214
7	11	49	7.071,068

2.3 Ẩn nội dung của một ô

Chúng ta có thể giấu đi nội dung của một ô bất kì, trong bảng ngang cũng như bảng dọc.

2.3.1 Ẩn giá trị

Để ẩn đi một giá trị (của biến) nào đó, trong danh sách các biến, chúng ta chỉ cần đặt trước giá trị đó kí tự "@". Trong ví dụ sau đây, chúng ta sẽ ẩn đi giá trị thứ 2 và thứ 5:

```
\htablecalc[3]{$x$}{x=-4,@-2,0,2.25,@7}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

x	-4		0	2.25	
$f(x) = 2x - 3$	-11	-7	-3	1.5	11
x^2	16	4	0	5.062,5	49
$h(x) = \sqrt{x^2 + 1}$	4.123,106	2.236,068	1	2.462,214	7.071,068

Nếu muốn thay @ bằng một kí tự khác thì chỉ cần dùng lệnh \def\tcnoshowmark, chẳng hạn nếu muốn thay @ bởi =, ta dùng \def\tcnoshowmark{=}.

2.3.2 Ẩn kết quả

Với mỗi giá trị cho trước, chúng ta có thể ẩn các kết quả thứ $a_1, a_2, \ldots, a_n$ bằng cách thêm $[a_1][a_2]\ldots[a_n]$ vào sau giá trị này, trong đó a_i được sắp xếp theo thứ tự tăng dần. Nếu một a_j nào đó bằng 0, tất cả các kết quả thứ a_k với $k > j$ đều được ẩn đi.Trong ví dụ sau đây, chúng ta sẽ:

- ẩn đi kết quả thứ 2 của giá trị đầu tiên với "-4[2]"

- hiển thị tất cả các kết quả của giá trị thứ 2 với "-2"
- ẩn đi kết quả thứ nhất và thứ 3 của giá trị thứ 3 với "0[1][3]"
- ẩn đi tất cả các kết quả của giá trị thứ 4 với "2.25[0]"
- ẩn đi tất cả các kết quả từ thứ 2 trở đi của giá trị thứ 5 với "7[2][0]"

```
\htablecalc[3]{$x$}{x=-4[2],-2,0[1][3],2.25[0],7[2][0]}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

x	-4	-2	0	2.25	7
$f(x) = 2x - 3$	-11	-7			11
x^2		4	0		
$h(x) = \sqrt{x^2 + 1}$	4.123,106	2.236,068			

Chúng ta cũng có thể kết hợp cú pháp này với @ để ẩn đi cả giá trị lẫn kết quả nếu muốn.

2.4 Chiều cao của hàng

Khi hiển thị một bảng, tabularcalc sẽ thực hiện lệnh \tcatbeginrow ở đầu của mỗi hàng. Theo mặc định, lệnh này được định nghĩa bởi \def\tcatbeginrow{\rule[-1.2ex]{0pt}{4ex}}. Mặc nhiên, lệnh này đã cố định chiều cao của tất cả các hàng bằng cách sử dụng các "cây thước" có bề rộng 0pt. Để dễ hình dung, đây là một "cây thước" với bề rộng 2pt nằm trước chữ "a" : a

Như vậy chúng ta có thể thay thế "cây thước" này bằng một thứ khác. Ví dụ :

```
\def\tcatbeginrow{%
    {\bfseries\number\tclin)\ }%
}
\htablecalc[3]{$x$}{x=-4,-2,0,2.25,7}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

0) x	-4	-2	0	2.25	7
1) $f(x) = 2x - 3$	-11	-7	-3	1.5	11
2) x^2	16	4	0	5.062,5	49
3) $h(x) = \sqrt{x^2 + 1}$	4.123,106	2.236,068	1	2.462,214	7.071,068

Trong ví dụ này, ở đầu hàng, chúng ta đã hiển thị (và in đậm) số thứ tự của mỗi hàng (số thứ tự này được cung cấp bởi bộ đếm \tclin).

2.5 Đường kẻ ngang

tabularcalc cung cấp 3 kiểu đường kẻ ngang, tương ứng với 3 tham số của lệnh \tcsethrule :

- "đường 0" là đường kẻ ngang ở trên cùng và dưới cùng của bảng;
- "đường 1" là đường kẻ ngang ngay dưới hàng thứ nhất;
- "các đường khác" là các đường kẻ ngang ngay dưới các hàng kết quả (trừ hàng cuối cùng).

Đây là cú pháp :
\tcsethrule{⟨*đường 0*⟩}{⟨*đường 1*⟩}{⟨*các đường khác*⟩}

Theo mặc định, cả ba tham số đều là \hline.

Sau đây là một ví dụ trong đó đường kẻ ngang dưới hàng thứ nhất là một đường đôi và các đường khác đều bị xoá :

```
\tcsethrule{\hline}{\hline\hline}{}
\htablecalc[3]{$x$}{x=-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

x	-2	-1	0	1	2	3
$2x$	-4	-2	0	2	4	6
$3x$	-6	-3	0	3	6	9
$4x$	-8	-4	0	4	8	12

Lệnh \tcresethrule cho phép trở lại các kiểu đường kẻ ngang như mặc định.

2.6 Tuỳ chọn đối với cột

2.6.1 Đường kẻ dọc

tabularcalc có hai loại cột : cột đầu tiên (bên trái) và các cột còn lại. Lệnh \setcoltype sử dụng một tham số tuỳ chọn và hai tham số bắt buộc :

- tham số tuỳ chọn, trống theo mặc định, xác định đường kẻ dọc "|" cuối cùng (bên phải) của bảng;
- tham số bắt buộc thứ nhất xác định các đường kẻ của cột thứ nhất (nếu không sử dụng \setcoltype thì nó được mặc định là "|c|";
- tham số bắt buộc thứ hai xác định các đường kẻ của các cột còn lại (nếu không sử dụng \setcoltype thì nó được mặc định là "c|"

Cú pháp :
\tcsetcoltype[⟨*tuỳ chọn*⟩]{⟨*tham số 1*⟩}{⟨*tham số 2*⟩}

Sau đây là một ví dụ :

```
\tcsetcoltype[||]{||c||}{c}
\htablecalc[3]{$x$}{x=-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

x	-2	-1	0	1	2	3
$2x$	-4	-2	0	2	4	6
$3x$	-6	-3	0	3	6	9
$4x$	-8	-4	0	4	8	12

Lệnh \tcresetcoltype cho phép trở lại với các kiểu cột như mặc định.

2.6.2 Độ rộng của cột

Thay vì sử dụng tham số cột "c" như trước đây, chúng ta có thể điều chỉnh độ rộng của tất cả các cột (trừ cột đầu tiên) bằng cách sử dụng tham số "m" của gói array. Trong ví dụ sau đây, chúng ta sẽ canh phải cột thứ nhất, các cột còn lại sẽ được canh giữa và có độ rộng 1.5 cm :

```
\usepackage{array}
\tcsetcoltype{|r|}{>{\centering\arraybackslash}m{1.5cm}|}
\htablecalc[3]{$x$}{x=-4,-2,0,2.25,7}
              {$f(x)=2x-3$}{2*x-3}
              {$x^2$}{x*x}
              {$h(x)=\sqrt{x^2+1}$}{round(root(2,x*x+1),6)}
```

x	-4	-2	0	2.25	7
$f(x) = 2x - 3$	-11	-7	-3	1.5	11
x^2	16	4	0	5.062,5	49
$h(x) = \sqrt{x^2 + 1}$	4.123,106	2.236,068	1	2.462,214	7.071,068

3 Nhập giá trị bằng công thức

Khi mà số lượng các giá trị của biến nhập vào khá lớn và tuân theo một quy luật toán học nào đó, thay vì nhập vào tất cả các giá trị đó, chúng ta chỉ cần nhập vào công thức của chúng. Ví dụ như cú pháp :

```
\htablecalc[2]{$x$}{x=-3,-1,1,3,5,7,9,11,13}
              {$2x$}{2*x}
              {$x^2$}{x*x}
```

có thể được thay thế bởi :

```
\htablecalc[2]{$x$}{x=a;a=-3:13[2]}
              {$2x$}{2*x}
              {$x^2$}{x*x}
```

Sự xuất hiện của dấu chấm phẩy `";"` đã thay đổi đã thay đổi cách đọc tham số chứa "danh sách" các giá trị của biến : ta thấy rằng ở phía bên phải của dấu chấm phẩy, biến trung gian `"a"` được gán các giá trị nguyên từ -3 đến 13 với gia số bằng 2 : các giá trị là các số *lẻ* liên tiếp; ở bên trái của dấu chấm phẩy, biến chính `x` được gán giá trị của biến trung gian `a` do đó nó cũng sẽ được gán các giá trị là các số nguyên lẻ từ -3 đến 13. Từ đây ta còn có nhiều cách viết khác, chẳng hạn `{x=2*a+1:a=-2:6}` hoặc `{x=2*a-3;a=0:8}` (gia số bằng 1 theo mặc định)

Khi sử dụng cách viết này, chúng ta không thể ẩn đi giá trị của các ô như đã gặp ở trang 4.

Với cú pháp này, tham số chứa các giá trị của biến được viết ở dạng sau :

⟨*biến 1*⟩`=`⟨*công thức*⟩`;`⟨*biến 2*⟩`=`⟨*min*⟩`:`⟨*max*⟩`[`⟨*gia số*⟩`]`

trong đó :

- ⟨*biến 1*⟩ là biến chính (chúng ta cần tính giá trị của các công thức theo biến này);
- ⟨*biến 2*⟩ là biến trung gian (biến được dùng để xác định các giá trị của biến chính), biến này bắt buộc phải khác ⟨*biến 1*⟩.
- ⟨*công thức*⟩ là công thức của ⟨*biến 1*⟩ theo ⟨*biến 2*⟩;
- ⟨*min*⟩`:`⟨*max*⟩ là đoạn (hay khoảng) mà ⟨*biến 2*⟩ thay đổi (⟨*min*⟩ không nhất thiết phải nhỏ hơn ⟨*max*⟩);
- ⟨*gia số*⟩ là gia số (của cấp số cộng theo ⟨*biến 2*⟩, số hạng đầu tiên là ⟨*min*⟩ và số hạng cuối phải nhỏ hơn hoặc bằng ⟨*max*⟩ nếu gia số dương và lớn hơn hoặc bằng ⟨*max*⟩ nếu gia số âm); gia số này phải khác 0 (bằng `1` theo mặc định).

Như chúng ta đã thấy, có nhiều cách khác nhau để nhập vào các giá trị của biến theo kiểu trên. Dưới đây là một ví dụ khác, nó sẽ nhập vào các giá trị `{0,1,2,3,4,5,6,7,8,9,10}` :

- `{z=x;x=0:10}` trong đó `z` là biến chính;
- `{n=2*a;a=0:5[0.5]}` trong đó `n` là biến chính;
- `{x=y/10;y=0:100[10]}` trong đó `x` là biến chính;

Chú ý : khoảng của biến trung gian và gia số cần phải tương thích với nhau : nếu $\langle min \rangle < \langle max \rangle$ thì gia số phải là số dương còn nếu $\langle min \rangle > \langle max \rangle$ thì gia số phải là số âm. Ví dụ như với tham số `0:10[-1]`, hệ thống sẽ báo lỗi !

Sau đây là một ví dụ sử dụng các hàm số lượng giác của `fp`:

```
\htablecalc[6]{$x$\ [deg]}{x=a;a=15:75[15]}
              {$\sin x$}{round(sin(x*pi/180),6)}
              {$\cos x$}{round(cos(x*pi/180),6)}
              {$\tan x$}{round(tan(x*pi/180),6)}
              {$\sin^2x$}{round(sin(x*pi/180)^2,6)}
              {$\cos^2x$}{round(cos(x*pi/180)^2,6)}
              {$\tan^2x$}{round(tan(x*pi/180)^2,6)}
```

x [deg]	15	30	45	60	75
$\sin x$	0.258,819	0.5	0.707,107	0.866,025	0.965,926
$\cos x$	0.965,926	0.866,025	0.707,107	0.5	0.258,819
$\tan x$	0.267,949	0.577,35	1	1.732,051	3.732,051
$\sin^2 x$	0.066,987	0.25	0.5	0.75	0.933,013
$\cos^2 x$	0.933,013	0.75	0.5	0.25	0.066,987
$\tan^2 x$	0.071,797	0.333,333	1	3	13.928,203

Còn bảng sau đây là các luỹ thừa cơ số 10 cùng logarit thập phân, căn bậc hai và nghịch đảo của chúng :

```
\htablecalc[3]{Luỹ\- thừ\-a cơ\- số\- 10}{x=round(10^n,4);n=-3:3}
              {Logarit thậ\-p phâ\-n}{ln(x)/ln(10)}
              {Cã\- bậ\-c hai}{round(root(2,x),3)}
              {Số\- đả\-o}{1/x}
```

Luỹ thừa cơ số 10	0.001	0.01	0.1	1	10	100	1,000
Logarit thập phân	−3	−2	−1	0	1	2	3
Căn bậc hai	0.032	0.1	0.316	1	3.162	10	31.623
Số đảo	1,000	100	10	1	0.1	0.01	0.001

4 Tuỳ chọn nâng cao

4.1 Thi hành lệnh trong một ô

Macro `\defcellcode` cho phép thi hành lệnh trong một ô, một cột hay một hàng. Các ô của một bảng được đánh số như sau :

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)

Chỉ số đầu — số thứ tự của hàng — là giá trị của bộ đếm `\tclin` còn chỉ số thứ hai — số thứ tự của cột — là giá trị của bộ đếm `\tccol`.

Sau đây là cú pháp :
`\defcellcode{`⟨*chỉ số 1*⟩`}{`⟨*chỉ số 2*⟩`}{`⟨*code*⟩`}`

trong đó :

- {⟨*chỉ số 1*⟩} là số thứ tự của hàng;
- {⟨*chỉ số 2*⟩} là số thứ tự của cột;
- {⟨*code*⟩} là mã sẽ được thực hiện khi hiển thị ô;
- Nếu không nhập {⟨*chỉ số 1*⟩} thì {⟨*code*⟩} sẽ được thi hành đối với tất cả các hàng;
- Nếu không nhập {⟨*chỉ số 2*⟩} thì {⟨*code*⟩} sẽ được thi hành đối với tất cả các cột;

Cần chú rằng {⟨*code*⟩} chỉ được thực hiện khi *hiển thị ô*, và lúc này, giá trị của bộ đếm \tccol không còn là số thứ tự của cột nữa, do đó chúng ta không thể sử dụng \tccol để thay thế cho {⟨*chỉ số 2*⟩} trong lệnh \defcellcode. Ngược lại, giá trị của bộ đếm \tclin chính là số thứ tự của hàng được hiển thị.
Nếu lệnh \defcellcode được thi hành nhiều lần với {⟨*code*⟩} khác nhau trên cùng một ô nào đó, thì chúng sẽ được thi hành theo thứ tự đã viết chúng.

Sau đây là một ví dụ, nội dung của ô (2 , 3) sẽ được in bằng màu xanh dương, hàng 1 sẽ được in bằng màu đỏ và cột 4 màu nâu.

```
\usepackage{color}
\defcellcode{2}{3}{\color{blue}}
\defcellcode{1}{}{\color{red}}
\defcellcode{}{4}{\color{brown}}
\htablecalc[3]{$x$}{x=-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

x	-2	-1	0	1	2	3
$2x$	-4	-2	0	2	4	6
$3x$	-6	-3	0	3	6	9
$4x$	-8	-4	0	4	8	12

Chúng ta thấy rằng số 2 của ô (1 , 4) có màu nâu. Thực ra đầu tiên nó đã được tô màu đỏ (xem dòng thứ 3 của khối lệnh trên) rồi sau đó mới được tô màu nâu (dòng 4).

Ngoài ra, chúng ta cũng có thể sử dụng \edefcellcode. Lúc này, {⟨*code*⟩} sẽ được thực hiện lần đầu tiên với \edef[1] khi xây dựng ô, và giá trị của bộ đếm \tccol chính là số thứ tự của ô đó. {⟨*code*⟩} sẽ được thực hiện thêm một lần nữa khi hiển thị ô.

Trong ví dụ sau đây, chúng ta sẽ tô màu xanh dương nội dung của tất cả các cột có số thứ tự lớn hơn 2 :

```
\usepackage{color}
\edefcellcode{}{}{%
    \ifnum\tccol>2 \noexpand\color{blue}\fi}
\htablecalc[3]{$x$}{x=-2,-1,0,1,2,3}
              {$2x$}{2*x}
              {$3x$}{3*x}
              {$4x$}{4*x}
```

x	-2	-1	0	1	2	3
$2x$	-4	-2	0	2	4	6
$3x$	-6	-3	0	3	6	9
$4x$	-8	-4	0	4	8	12

[1] Có thể thêm vào \noexpand ở trước lệnh mà ta không muốn thi hành lúc này.

4.2 Tuỳ chọn hiển thị

4.2.1 Các macro \printvalue và \printresult

Để in ra một giá trị, lệnh \tcprintvalue sẽ được gọi. Nó chỉ có một tham số là số chữ số thập phân cần in ra (được cho bởi fp).
Theo mặc định, \tcprintvalue được định nghĩa như sau :

```
\def\tcprintvalue#1{\numprint{#1}}
```

Ta thấy rằng lệnh \numprint được gọi để in ra đẹp hơn.

Để in ra một kết quả, lệnh \tcprintresult sẽ được gọi. Lệnh này có **hai** tham số : một là kết quả cho bởi fp và hai là giá trị của biến.
Theo mặc định, \tcprintresult được định nghĩa như sau :

```
\def\tcprintresult#1#2{\numprint{#1}}
```

Như vậy ta thấy rằng tham số #2 (giá trị của biến) không phụ thuộc vào \tcprintresult. Tuy nhiên ta cũng có thể thay đổi điều này. Trong ví dụ sau đây, chúng ta sẽ in ra một chữ X màu đỏ khi cạnh của hình vuông (chính là tham số #2) là một số âm, nếu không ta sẽ in ra kết quả (diện tích hình vuông, tham số #1) cùng với đơn vị. Hơn nữa là sẽ tô màu xanh tất cả các kết quả nhỏ hơn 10 :

```
\usepackage{color}
\def\tcprintresult#1#2{%
    \ifdim#1pt<10pt\color{blue}\fi
    \ifdim#2pt<0pt
        \color{red}\texttt{X}%
    \else
        \numprint[cm^2]{#1}%
    \fi}
\htablecalc{Cạ\-nh củ\-a hì\-nh vuô\-ng}{x=0.7,-10,3,-2,5,12}
           {Diệ\-n tí\-ch}{x*x}
```

Cạnh của hình vuông	0.7	−10	3	−2	5	12
Diện tích	0.49 cm^2	X	9 cm^2	X	25 cm^2	144 cm^2

4.2.2 Điều chỉnh các số làm tròn

Các phép tính được tính bởi fp có độ chính xác rất cao, đến 12 chữ số thập phân. Ví dụ phép tính $\sqrt{10}$ bởi fp sẽ cho ra kết quả :

3.162,277,660,168,379,312

11 chữ số thập phân đầu tiên là chính xác, chữ số thứ 12 là làm tròn.

Đối với các kết quả, chúng ta có thể sử dụng hàm round(kết quả,số chữ số thập phần) của fp (như đã thấy trong các ví dụ trước). Nhưng nếu muốn đơn giản hoá các câu lệnh chúng ta cũng có thể sử dụng lệnh \tcprintroundresult của tabularcalc. Tham số của nó là số chữ số thập phân mà ta cần in ra. Nếu thay \tcprintroundresult bởi \tcprintroundresult* thì nó sẽ thêm vào (nếu cần) các kết quả chẵn (chính xác) một số chữ số 0 cho đủ số chữ số thập phân cần in ra. Nếu tham số được để trống thì tabularcalc sẽ không làm tròn kết quả.

```
\tcprintroundresult{3}
\htablecalc{$x$}{x=2,3,4,5}
           {$\sqrt{x}$}{root(2,x)}
```

x	2	3	4	5
$\sqrt{x}$	1.414	1.732	2	2.236

```
\tcprintroundresult*{3}
\htablecalc{$x$}{x=2,3,4,5}
           {$\sqrt{x}$}{root(2,x)}
```

x	2	3	4	5
$\sqrt{x}$	1.414	1.732	2.000	2.236

Lưu ý : khi nhập giá trị bằng công thức (xem trang 7), **không nên** sử dụng hàm số round (trong công thức liên hệ giữa biến chính và biến trung gian), bởi vì nó sẽ làm tròn các giá trị của biến chính và do đó các kết quả sẽ không chính xác. Ví dụ :

```
\htablecalc{Cã\-n bậ\-c hai}{x=round(root(2,k),2);k=2:4}
            {Bi\-nh phươ\-\-ng}{x*x}
```

Căn bậc hai	1.41	1.73	2
Bình phương	1.988,1	2.992,9	4

Ta thấy rằng các kết quả đã không còn chính xác. Tốt hơn hết là nên sử dụng \tcprintroundvalue và \tcprintroundresult với cùng một tham số :

```
\tcprintroundvalue{2}
\htablecalc{Cã\-n bậ\-c hai}{x=root(2,k);k=2:4}
            {Bi\-nh phươ\-\-ng}{x*x}
```

Căn bậc hai	1.41	1.73	2
Bình phương	1.999,999,999,999,999,98	2.999,999,999,999,999,935	3.999,999,999,999,999,96

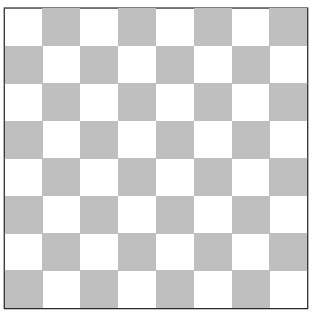
Các kết quả, không được làm tròn, là rất gần với các kết quả chính xác.

4.2.3 Một chút thư giãn

Chúng ta sẽ dùng tabularcalc để vẽ một bàn cờ vua trong đó mỗi ô có cạnh bằng 0.5 cm :

- dòng 1 cho bằng 0pt các phân cách của bảng để không ảnh hưởng đến chiều rộng 0.5 cm mà ta sẽ vẽ ở các dòng tiếp theo ngay dưới đây;
- dòng 2 cho phép không in ra bất kì giá trị hay kết quả nào;
- dòng 3 cho phép chỉ có hai đường kẻ ngang là đường trên cùng và dưới cùng của bảng, còn dòng 4 là đường bên trái và bên phải cùng của bảng;
- dòng 5 thiết lập một "cây thước" cao 0.5 cm ở mỗi hàng của bảng;
- dòng 7 kiểm tra tính chẵn lẻ của tổng số thứ tự hàng + số thứ tự cột của mỗi ô, và nếu tổng này là lẻ thì ta sẽ tô màu xám các ô này ở dòng 8.

```
\arraycolsep=0pt\tabcolsep=0pt
\def\tcprintvalue#1{}\def\tcprintresult#1#2{}
\tcsethrule{\hline}{}{}
\tcsetcoltype[|]{|m{0.5cm}}{m{0.5cm}}
\def\tcatbeginrow{\rule[-0.2cm]{0pt}{0.3cm}}
\edefcellcode{}{}{%
    \ifodd\numexpr\tccol+\tclin\relax
        \noexpand\cellcolor{lightgray}\fi
}
\htablecalc[7]{}{x=1,2,3,4,5,6,7}
    {}{x}{}{x}{}{x}{}{x}{}{x}{}{x}{}{x}
```



5 Xuất một bảng ra một tập tin

Nếu lệnh `\tcwritetofile{`⟨*filename*⟩`}` (với một tham số bắt buộc là tên tập tin không chứa phần mở rộng) đặt trước `\htablecalc` hoặc `\vtablecalc` thì hai lệnh này sẽ không in ra bảng như thường lệ mà chúng sẽ tạo ra một tập tin ⟨*filename*⟩`.tex` (trong cùng thư mục với tập tin tạo ra nó) chứa mã TEX của bảng.

Sau đây là một ví dụ :

```
\tcwritetofile{mytable}
\defcellcode{}{2}{\color{blue}}
\htablecalc[2]{$x$}{x=k;k=0:4}
                {$2x$}{2*x}
                {$x^2$}{x*x}
\tcresetcellcode
```

Một tập tin có tên `mytable.tex` sẽ được tạo ra trong cùng thư mục với tập tin đang được sử dụng và nó chứa mã TEX của bảng trên :

```
\begin {tabular}{|c|*{5}{c|}}\hline
\tcatbeginrow $x$&\tcprintvalue {0}&\color {blue}\tcprintvalue {1}&\
     tcprintvalue {2}&\tcprintvalue {3}&\tcprintvalue {4}\\\hline
\tcatbeginrow $2x$&\tcprintresult {0}{0}&\color {blue}\tcprintresult
     {2}{1}&\tcprintresult {4}{2}&\tcprintresult {6}{3}&\tcprintresult
      {8}{4}\\ \hline
\tcatbeginrow $x^2$&\tcprintresult {0}{0}&\color {blue}\tcprintresult
     {1}{1}&\tcprintresult {4}{2}&\tcprintresult {9}{3}&\tcprintresult
      {16}{4}\\ \hline
\end {tabular}
```

Điều này giúp người sử dụng dễ dàng thay đổi các đặc điểm của bảng theo ý muốn và sau đó có thể chèn tập tin này vào tập tin của một tài liệu nào đó với lệnh :

```
\input{mytable.tex}
```

và đây là kết quả :

x	0	1	2	3	4
$2x$	0	2	4	6	8
x^2	0	1	4	9	16

6 Sử dụng khái niệm "trung tố" hay "hậu tố"

Chúng ta có thể sử dụng cả hai khái niệm "trung tố" (infix) và "hậu tố" (postfix) bởi vì tabularcalc sử dụng `\FPeval` và với `\FPeval` thì cả hai cách dùng này đều được chấp nhận. Trong ví dụ sau đây, để tạo ra cùng một bảng, ta sẽ sử dụng khái niệm "trung tố" sau đó là "hậu tố". Hiển nhiên kết quả nhận được là như nhau bởi vì ta sử dụng cùng một "máy tính" :

```
\tcprintroundvalue{6}
\tcprintroundresult{6}
Vở\-i khá\-i niệ\-m trung tố\-\par
\htablecalc[3]{$x=10^k$ o\‘u $k\in[-3;3]$}{x=10^k;k=-3:3}
                {$\log x$}{ln(x)/ln(10)}
                {$\sqrt{x}$}{root(2,x)}
                {$\frac{1}{x}$}{1/x}

\medskip
Vở\-i khá\-i niệ\-m hậ\-u tố\-\par
\htablecalc[3]{$x=10^k$ o\‘u $k\in[-3;3]$}{x=k 10 pow;k=-3:3}
                {$\log x$}{x ln 10 ln div}
                {$\sqrt{x}$}{2 x root}
                {$\frac{1}{x}$}{1 x div}
```

Với khái niệm trung tố

$x = 10^k$ où $k \in [-3;3]$	0.001	0.01	0.1	1	10	100	1,000
$\log x$	-3	-2	-1	0	1	2	3
$\sqrt{x}$	0.031,623	0.1	0.316,228	1	3.162,278	10	31.622,777
$\frac{1}{x}$	1,000	100	10	1	0.1	0.01	0.001

Với khái niệm hậu tố

$x = 10^k$ où $k \in [-3;3]$	0.001	0.01	0.1	1	10	100	1,000
$\log x$	-3	-2	-1	0	1	2	3
$\sqrt{x}$	0.031,623	0.1	0.316,228	1	3.162,278	10	31.622,777
$\frac{1}{x}$	1,000	100	10	1	0.1	0.01	0.001

Nếu như bạn thông thạo cả hai cách viết thì khái niệm hậu tố có lẽ là tốt hơn vì nó sẽ giúp tính toán nhanh hơn. Chẳng hạn biểu thức $\cos x(1 - \cos x)$ sẽ được viết như sau với khái niệm trung tố

```
cos(x)*(1-cos(x))
```

Như vậy sẽ phải mất hai lần tính $\cos x$.

Còn với khái niệm hậu tố thì $\cos x$ chỉ được tính một lần :

```
x cos copy 1 swap sub mul
```

⋆

⋆ ⋆

Đó là tất cả, hi vọng rằng gói này sẽ giúp ích cho các bạn !
Mọi ý kiến đóng góp xin gửi về cho tác giả qua email.

Christian Tellechea

Nd : nếu bạn muốn đóng góp ý kiến cho bản dịch, xin gửi email đến địa chỉ huudienkhue.le@gmail.com.
Xin cảm ơn !