The vertical (or) bar separates required arguments from optional arguments in function declarations. The bar may appear at most once.

### Description

Mata functions may have various numbers of arguments. How you write programs that allow these optional arguments is described below.

### Remarks and examples

Remarks are presented under the following headings:

- What are optional arguments?
- How to code optional arguments
- Examples revisited

#### What are optional arguments?

**Example 1**

You write a function named `ditty()`. Function `ditty()` allows the caller to specify two or three arguments:

```plaintext
real matrix  ditty(real matrix A, real matrix B, real scalar scale)
real matrix  ditty(real matrix A, real matrix B)
```

If the caller specifies only two arguments, results are as if the caller had specified the third argument equal to missing; that is, `ditty(A, B)` is equivalent to `ditty(A, B, .)`
Example 2

You write function `gash()`. Function `gash()` allows the caller to specify one or two arguments:

```plaintext
real matrix gash(real matrix A, real matrix B)
real matrix gash(real matrix A)
```

If the caller specifies only one argument, results are as if `J(0,0,. )` were specified for the second.

Example 3

You write function `easygoing()`. Function `easygoing()` takes three arguments but allows the caller to specify three, two, one, or even no arguments:

```plaintext
real scalar easygoing(real matrix A, real matrix B, real scalar scale)
real scalar easygoing(real matrix A, real matrix B)
real scalar easygoing(real matrix A)
real scalar easygoing()
```

If `scale` is not specified, results are as if `scale = 1` were specified. If `B` is not specified, results are as if `B = A` were specified. If `A` is not specified, results are as if `A = I(2)` were specified.

Example 4

You write function `midsection()`. `midsection()` takes three arguments, but users may specify only two—the first and last—if they wish.

```plaintext
real matrix midsection(real matrix A, real vector w, real matrix B)
real matrix midsection(real matrix A, real matrix B)
```

If `w` is not specified, results are as if `w = J(1,cols(A),1)` was specified.

How to code optional arguments

When you code

```plaintext
function nebulous(a, b, c)
{
    ...
}
```

you are stating that function `nebulous()` requires three arguments. If the caller specifies fewer or more, execution will abort.
If you code

```c
function nebulous(a, b, |c)
{
    ...
}
```

you are stating that the last argument is optional. Note the vertical or bar in front of `c`.

If you code

```c
function nebulous(a, |b, c)
{
    ...
}
```

you are stating that the last two arguments are optional; the user may specify one, two, or three arguments.

If you code

```c
function nebulous(|a, b, c)
{
    ...
}
```

you are stating that all arguments are optional; the user may specify zero, one, two, or three arguments.
The arguments that the user does not specify will be filled in according to the arguments’ type,

<table>
<thead>
<tr>
<th>If the argument type is ...</th>
<th>The default value will be ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>undeclared</td>
<td>J(0, 0, .)</td>
</tr>
<tr>
<td>transmorphic matrix</td>
<td>J(0, 0, .)</td>
</tr>
<tr>
<td>real matrix</td>
<td>J(0, 0, .)</td>
</tr>
<tr>
<td>complex matrix</td>
<td>J(0, 0, 1i)</td>
</tr>
<tr>
<td>string matrix</td>
<td>J(0, 0, &quot;&quot;)</td>
</tr>
<tr>
<td>pointer matrix</td>
<td>J(0, 0, NULL)</td>
</tr>
<tr>
<td>transmorphic rowvector</td>
<td>J(1, 0, .)</td>
</tr>
<tr>
<td>real rowvector</td>
<td>J(1, 0, .)</td>
</tr>
<tr>
<td>complex rowvector</td>
<td>J(1, 0, 1i)</td>
</tr>
<tr>
<td>string rowvector</td>
<td>J(1, 0, &quot;&quot;)</td>
</tr>
<tr>
<td>pointer rowvector</td>
<td>J(1, 0, NULL)</td>
</tr>
<tr>
<td>transmorphic colvector</td>
<td>J(0, 1, .)</td>
</tr>
<tr>
<td>real colvector</td>
<td>J(0, 1, .)</td>
</tr>
<tr>
<td>complex colvector</td>
<td>J(0, 1, 1i)</td>
</tr>
<tr>
<td>string colvector</td>
<td>J(0, 1, &quot;&quot;)</td>
</tr>
<tr>
<td>pointer colvector</td>
<td>J(0, 1, NULL)</td>
</tr>
<tr>
<td>transmorphic vector</td>
<td>J(1, 0, .)</td>
</tr>
<tr>
<td>real vector</td>
<td>J(1, 0, .)</td>
</tr>
<tr>
<td>complex vector</td>
<td>J(1, 0, 1i)</td>
</tr>
<tr>
<td>string vector</td>
<td>J(1, 0, &quot;&quot;)</td>
</tr>
<tr>
<td>pointer vector</td>
<td>J(1, 0, NULL)</td>
</tr>
<tr>
<td>transmorphic scalar</td>
<td>J(1, 1, .)</td>
</tr>
<tr>
<td>real scalar</td>
<td>J(1, 1, .)</td>
</tr>
<tr>
<td>complex scalar</td>
<td>J(1, 1, C(.))</td>
</tr>
<tr>
<td>string scalar</td>
<td>J(1, 1, &quot;)</td>
</tr>
<tr>
<td>pointer scalar</td>
<td>J(1, 1, NULL)</td>
</tr>
</tbody>
</table>

Also, the function `args()` (see [M-5] args()) will return the number of arguments that the user specified.

The vertical bar can be specified only once. That is sufficient, as we will show.

**Examples revisited**

**Example 1**

In this example, real matrix function `ditty(A, B, scale)` allowed real scalar `scale` to be optional. If `scale` was not specified, results were as if `scale=.` had been specified. This can be coded

```plaintext
real matrix ditty(real matrix A, real matrix B, |real scalar scale) {
  ...
}
```
The body of the code is written just as if \textit{scale} were not optional because, if the caller does not specify the argument, the missing argument is automatically filled in with missing, per the table above.

\section*{Example 2}

Real matrix function \texttt{gash}(A, B) allowed real matrix \(B\) to be optional, and if not specified, \(B = J(0,0,\ldots)\) was assumed. Hence, this is coded just as example 1 was coded:

\begin{verbatim}
real matrix gash(real matrix A, |real matrix B)
{
...
}
\end{verbatim}

\section*{Example 3}

Real scalar function \texttt{easygoing}(A, B, \textit{scale}) allowed all arguments to be optional. \textit{scale} = 1 was assumed, \(B = A\), and if necessary, \(A = I(2)\).

\begin{verbatim}
real scalar easygoing(|real matrix A, real matrix B, real scalar scale)
{
...
if (args()==2) scale = 1
else if (args==1) {
  B = A
  scale = 1
}
else if (args()==0) {
  A = B = I(2)
  scale = 1 ;
}
...
}
\end{verbatim}

\section*{Example 4}

Real matrix function \texttt{midsection}(A, \(w\), B) allowed \(w\)—its middle argument—to be omitted. If \(w\) was not specified, \(J(1, \text{cols}(A), 1)\) was assumed. Here is one solution:

\begin{verbatim}
real matrix midsection(a1, a2, |a3)
{
  if (args()==3) return(midsection_u(a1, a2, a3))
  else return(midsection_u(a1, J(1,cols(a1),1), a2))
}
real matrix midsection_u(real matrix A, real vector w, real matrix B)
{
  ...
}
\end{verbatim}
We will never tell callers about the existence of midsection_u() even though midsection_u() is our real program.

What we did above was write midsection() to take two or three arguments, and then we called midsection_u() with the arguments in the correct position.

Also see

[M-2] intro — Language definition