for

for <*variable name*> [=] <*n1*>, <*n2*>, [<*step size*>]...	***for*** uses a special syntax. Specify a variable that is to take the values starting with *n1* and ending with *n2* in an incremented loop. You can specify the amount by which the variable is incremented using *step size* (this is optional).

You can make Semper repeat a group of commands by surrounding them with the commands **for...loop**.

Examples

```
for n=11,14; ps n to n+10
section; survey; loop
```

This sequence of commands produces in pictures 21 to 24 the rotational averages of the power spectra of pictures 11 to 14, and types their ranges.

```
for x 1,0, -0.25; type x; loop
```

This sequence of commands types 1, 0.75, 0.5, 0.25 and 0 in turn on successive lines.

Description

You end a **for** loop by specifying the **loop** command. If you omit a step size for the loop the default step is 1 or –1 according to the loop direction. Note that you cannot specify a zero step size.

You can extend a **for...loop** over several lines, as in the first command example. If you are working interactively, the terminal prompt reminds you about active loops. There is also an installation dependent maximum loop length, typically 1000 characters. This limit applies in *run* files, but not in *library* programs.

The loop variable is local to its **for** loop, that is, its original state or value is restored when the loop ends. (See the **local** command for further detail).

You can *nest* **for** loops up to an installation dependent maximum depth, which is typically 6. Each **for** command in a nested loop requires a corresponding **loop** command. Note that you can specify a variable name with **loop** to clarify your structure. For example:

```
for s=1,2
for n=5,8; type n; loop n
type 'who do we appreciate?...Semper 6!'
loop s
```

for

You can also re-use the same loop variable in each nested loop as each variable is *local* to its loop.

To break out of a **for...loop** use the command **break** or **next**.

Notes

restrictions: a special restriction applies to **for** loops with respect to numbered macros (which will not be supported in later increases of Semper). Loops are not executed correctly in a numbered macro when the macro itself is called from within a library program.

see also: **break, loop, next**