Followed by a view command to see where we are:

```
DB<1> v
7:              my ($deg, $num) = ($1, $2);
8:              my ($in, $out) = ($num, $num);
9:              $DB::single=2;
10==>           if ($deg eq 'c') {
11:                     $deg = 'f';
12:                     $out = &c2f($num);
13              } else {
14:                     $deg = 'c';
15:                     $out = &f2c($num);
16              }
```

And a print to show what values we're currently using:

```
DB<1> p $deg, $num
f33.3
```

We can put another break point on any line beginning with a colon, we'll use line 17 as that's just as we come out of the subroutine, and we'd like to pause there later on:

```
DB<2> b 17
```

There's no feedback from this, but you can see what breakpoints are set by using the list 'L' command:

```
DB<3> L
temp:
        17:            print "$out $deg\n";
        break if (1)
```

Note that to delete a breakpoint you use 'd' or 'D'.

Now we'll continue down into our subroutine, this time rather than by line number, we'll use the subroutine name, followed by the now familiar 'v':

```
DB<3> c f2c
main::f2c(temp:30):             my $f = shift;

DB<4> v
24:     exit;
25
26      sub f2c {
27==>           my $f = shift;
28:             my $c = 5 * $f - 32 / 9;
29:             return $c;
30      }
31
32      sub c2f {
33:             my $c = shift;
```

Note that if there was a subroutine call between us and line 29, and we wanted to **single-step** through it, we could use the '**s**' command, and to step over it we would use '**n**' which would execute the sub, but not descend into it for inspection. In this case though, we simply continue down to line 29:

```
DB<4> c 29
main::f2c(temp:29):             return $c;
```