[7.9] Passing optional parameters to functions and programs

Many programming languages support passing optional parameters to functions and programs, but TI BASIC does not. For example, you might have a function *f1()* that takes three or four parameters:

```
f1(p1,p2,p3,p4)
```

but if the user doesn't supply a value for the third parameter:

```
f1(p1,p2,,p4)
```

then the program supplies a default value for *p3*, or takes some other action. Note that this feature is supported in some built-in 89/92+ functions, such as *LinReg(),* for example.

You can simulate this operation to some extent in your own programs by passing the parameters as lists. For the example above, the call would be

```
f1(p1,p2,{p3},p4)
```

The *p3* parameter is always included in the call, but it may be an emply list. This can be determined if dim(p3) = 0. There are other variations on this theme, for example

```
f1(p1,{p2, p3, p4})
```

can be used to supply up to three optional parameters. Again *dim()* is used to determine how many parameters were supplied.

Alex Astashyn points out that there are some disadvantages to this method:

- You have to write additional code to recognize which arguments are supplied and to provide defaults.
- You have to remember to enter additional arguments in the list.
- You have to know what the default values are, to decide if you need to change them.
- You don't want to put the burden of remembering all this on the user, so it's easier to obligate the user to use the fixed number of arguments.

On the other hand, Frank Westlake notes these advantages:

- You can write one function instead of several very similar functions, which saves memory.
- The user doesn't have to remember which function to use for a given case.
- You can use an empty list as a request for help. If the list has no elements, the program returns a string that describes the list format.

(Credit to Glenn Fisher, Alex Astashyn, and Frank Westlake)