For Every "Thing," Call the Callback and Take Action

```
for (dl = head->next; head != dl;
     dl = dl->next) {

  result = (*func) (dl, arg);

  switch (result) {
```

Loop over all things.

Call the callback function.

Take action based on callback's response.

41

Cases to Return "Thing" After Possibly Removing It

```
switch (result) {
    case DL_REMOVE_AND_STOP:
        dl_remove (dl);
    case DL_STOP_AND_RETURN:
        return dl;
```

Remove the "thing"…

… and return it! (no **break**)

This case returns "thing" without removing.

42

Cases to Remove "Thing" and Continue (Maybe Freeing)

```
case DL_REMOVE_AND_CONTINUE:
case DL_FREE_AND_CONTINUE:
    remove = dl;
    dl = dl->prev;
    dl_remove (remove);
    if (DL_FREE_AND_CONTINUE ==
        result) {
        free (remove);
    }
    break;
```

Copy "thing" to **remove**.

Loop update reads this element's **next**.

43

Cases to Remove "Thing" and Continue (Maybe Freeing)

```
case DL_REMOVE_AND_CONTINUE:
case DL_FREE_AND_CONTINUE:
    remove = dl;
    dl = dl->prev;
    dl_remove (remove);
    if (DL_FREE_AND_CONTINUE ==
        result) {
        free (remove);
    }
    break;
```

Remove "thing" from list.

If requested, free "thing" (other data can be freed by callback).

44