```
 1: // $Id: listfree.cpp,v 1.34 2022-01-14 19:52:05-08 - - $
 2:
 3: // Show how to break a cycle in a simple circular list.
 4:
 5: #include <algorithm>
 6: #include <iostream>
 7: #include <memory>
 8: using namespace std;
 9:
10: struct node;
11:
12: using node_ptr = shared_ptr<node>;
13:
14: template <typename Type>
15: ostream& operator<< (ostream& out, shared_ptr<Type> ptr) {
16:    return out << "{" << ptr.get() << "," << ptr.use_count() << "}";
17: }
18:
19: struct node {
20:    int value {};
21:    node_ptr link;
22:    void show (const char* name);
23:    node (int value_, node_ptr link_): value(value_), link(link_) {
24:       show (__PRETTY_FUNCTION__); cout << endl;
25:    }
26:    ~node() { show (__PRETTY_FUNCTION__); cout << endl; }
27: };
28:
29: void node::show (const char* name) {
30:    cout << name << ": " << this << "->node(" << value
31:         << "," << link << ")";
32: }
33:
34: int main (int argc, char** argv) {
35:    cout << "Command:";
36:    for_each (&argv[0], &argv[argc], [](char* arg){cout << " " << arg;});
37:    cout << endl;
38:    bool break_cycle = argc > 1 and argv[1] == "-f"s;
39:    node_ptr list = make_shared<node> (1,
40:                    make_shared<node> (2,
41:                    make_shared<node> (3, nullptr)));
42:    list->link->link->link = list;
43:    cout << "list = " << list << endl;
44:    for (auto curr = list;;) {
45:       cout << curr << " -> "; curr->show ("curr"); cout << endl;
46:       curr = curr->link;
47:       if (curr == list) break;
48:    }
49:    if (break_cycle) list->link = nullptr;
50:    cout << __PRETTY_FUNCTION__ << ": return 0;" << endl;
51:    return 0;
52: }
53:
54: //TEST// valgrind listfree -0 >listfree.out-0 2>&1
55: //TEST// valgrind listfree -f >listfree.out-f 2>&1
56: //TEST// mkpspdf listfree.ps listfree.cpp listfree.out-*
57:
```

```
    1: ==26766== Memcheck, a memory error detector
    2: ==26766== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al
.
    3: ==26766== Using Valgrind-3.17.0 and LibVEX; rerun with -h for copyright
info
    4: ==26766== Command: listfree -0
    5: ==26766==
    6: Command: listfree -0
    7: node::node(int, node_ptr): 0x5c450b0->node(3,{0,0})
    8: node::node(int, node_ptr): 0x5c45120->node(2,{0x5c450b0,3})
    9: node::node(int, node_ptr): 0x5c45190->node(1,{0x5c45120,3})
   10: list = {0x5c45190,3}
   11: {0x5c45190,4} -> curr: 0x5c45190->node(1,{0x5c45120,2})
   12: {0x5c45120,3} -> curr: 0x5c45120->node(2,{0x5c450b0,2})
   13: {0x5c450b0,3} -> curr: 0x5c450b0->node(3,{0x5c45190,3})
   14: int main(int, char**): return 0;
   15: ==26766==
   16: ==26766== HEAP SUMMARY:
   17: ==26766==     in use at exit: 120 bytes in 3 blocks
   18: ==26766==   total heap usage: 4 allocs, 1 frees, 147 bytes allocated
   19: ==26766==
   20: ==26766== LEAK SUMMARY:
   21: ==26766==    definitely lost: 40 bytes in 1 blocks
   22: ==26766==    indirectly lost: 80 bytes in 2 blocks
   23: ==26766==      possibly lost: 0 bytes in 0 blocks
   24: ==26766==    still reachable: 0 bytes in 0 blocks
   25: ==26766==         suppressed: 0 bytes in 0 blocks
   26: ==26766== Rerun with --leak-check=full to see details of leaked memory
   27: ==26766==
   28: ==26766== For lists of detected and suppressed errors, rerun with: -s
   29: ==26766== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
    1: ==26771== Memcheck, a memory error detector
    2: ==26771== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al
.
    3: ==26771== Using Valgrind-3.17.0 and LibVEX; rerun with -h for copyright
info
    4: ==26771== Command: listfree -f
    5: ==26771==
    6: Command: listfree -f
    7: node::node(int, node_ptr): 0x5c450b0->node(3,{0,0})
    8: node::node(int, node_ptr): 0x5c45120->node(2,{0x5c450b0,3})
    9: node::node(int, node_ptr): 0x5c45190->node(1,{0x5c45120,3})
   10: list = {0x5c45190,3}
   11: {0x5c45190,4} -> curr: 0x5c45190->node(1,{0x5c45120,2})
   12: {0x5c45120,3} -> curr: 0x5c45120->node(2,{0x5c450b0,2})
   13: {0x5c450b0,3} -> curr: 0x5c450b0->node(3,{0x5c45190,3})
   14: node::~node(): 0x5c45120->node(2,{0x5c450b0,2})
   15: node::~node(): 0x5c450b0->node(3,{0x5c45190,3})
   16: int main(int, char**): return 0;
   17: node::~node(): 0x5c45190->node(1,{0,0})
   18: ==26771==
   19: ==26771== HEAP SUMMARY:
   20: ==26771==     in use at exit: 0 bytes in 0 blocks
   21: ==26771==   total heap usage: 4 allocs, 4 frees, 147 bytes allocated
   22: ==26771==
   23: ==26771== All heap blocks were freed -- no leaks are possible
   24: ==26771==
   25: ==26771== For lists of detected and suppressed errors, rerun with: -s
   26: ==26771== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```