

```
1: // $Id: insertlist.cpp,v 1.42 2014-04-24 14:10:53-07 - - $
2:
3: //
4: // List insertion algorithm.
5: // Insert nodes into a singly-linked list using only operator<
6: // to form comparisons. Do not insert elements that already
7: // exist.
8: //
9:
10: #include <iostream>
11: #include <stdexcept>
12: #include <string>
13: using namespace std;
14:
15: template <typename Type>
16: struct xless {
17:     bool operator() (const Type& left, const Type& right) const {
18:         return left < right;
19:     }
20: };
21:
22: template <typename Type>
23: struct xgreater {
24:     bool operator() (const Type& left, const Type& right) const {
25:         return left > right;
26:     }
27: };
28:
```

```
29:
30: template <typename element, class Less=xless<element>>
31: struct linked_list {
32:     struct node {
33:         element elt;
34:         node* link;
35:         node (const element& elt_, node* link_): elt(elt_), link(link_) {}
36:     };
37:     Less less;
38:     node* head = nullptr;
39:
40:     linked_list(){} // Needed because default is suppressed.
41:     linked_list (const linked_list&) = delete;
42:     linked_list& operator= (const linked_list&) = delete;
43:
44:     void insert_ascending (const element& newelt) {
45:         node** curr = &head;
46:         while (*curr != nullptr and less ((*curr)->elt, newelt)) {
47:             curr = &(*curr)->link;
48:         }
49:         if (*curr == nullptr or less (newelt, (*curr)->elt)) {
50:             *curr = new node (newelt, *curr);
51:         }
52:     }
53:
54:     element& front() { return head->elt; }
55:
56:     void pop_front() {
57:         if (head == nullptr) {
58:             throw underflow_error ("linked_list<>::pop_front()");
59:         }
60:         node* old = head;
61:         head = head->link;
62:         delete old;
63:     }
64: };
65:
```

```
66:
67: template <typename element, class Less=xless<element>>
68: void process (int argc, char** argv, const string& label) {
69:     linked_list<string,Less> list;
70:     for (char** argp = &argv[1]; argp != &argv[argc]; ++argp) {
71:         cout << label << ": Insert: " << *argp << endl;
72:         list.insert_ascending (*argp);
73:     }
74:     while (list.head != nullptr) {
75:         cout << label << ": Sorted: " << list.front() << endl;
76:         list.pop_front();
77:     }
78:     cout << endl;
79: }
80:
81: int main (int argc, char** argv) {
82:     process<string> (argc, argv, "Default");
83:     process<string,xgreater<string>> (argc, argv, "Greater");
84:     return 0;
85: }
86:
87: /*
88: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
89: //TEST// grind --log-file=insertlist.out.log \
90: //TEST//     insertlist foo bar baz qux quux zxcvbnm asdfg qwerty \
91: //TEST//     bar baz foo quux qwerty hello hello 1234567890 \
92: //TEST//     >insertlist.out 2>&1
93: //TEST// mkpspdf insertlist.ps insertlist.cpp* insertlist.out*
94: */
```

[illegible]

```
1: Default: Insert: foo
2: Default: Insert: bar
3: Default: Insert: baz
4: Default: Insert: qux
5: Default: Insert: quux
6: Default: Insert: zxcvbnm
7: Default: Insert: asdfg
8: Default: Insert: qwerty
9: Default: Insert: bar
10: Default: Insert: baz
11: Default: Insert: foo
12: Default: Insert: quux
13: Default: Insert: qwerty
14: Default: Insert: hello
15: Default: Insert: hello
16: Default: Insert: 1234567890
17: Default: Sorted: 1234567890
18: Default: Sorted: asdfg
19: Default: Sorted: bar
20: Default: Sorted: baz
21: Default: Sorted: foo
22: Default: Sorted: hello
23: Default: Sorted: quux
24: Default: Sorted: qux
25: Default: Sorted: qwerty
26: Default: Sorted: zxcvbnm
27:
28: Greater: Insert: foo
29: Greater: Insert: bar
30: Greater: Insert: baz
31: Greater: Insert: qux
32: Greater: Insert: quux
33: Greater: Insert: zxcvbnm
34: Greater: Insert: asdfg
35: Greater: Insert: qwerty
36: Greater: Insert: bar
37: Greater: Insert: baz
38: Greater: Insert: foo
39: Greater: Insert: quux
40: Greater: Insert: qwerty
41: Greater: Insert: hello
42: Greater: Insert: hello
43: Greater: Insert: 1234567890
44: Greater: Sorted: zxcvbnm
45: Greater: Sorted: qwerty
46: Greater: Sorted: qux
47: Greater: Sorted: quux
48: Greater: Sorted: hello
49: Greater: Sorted: foo
50: Greater: Sorted: baz
51: Greater: Sorted: bar
52: Greater: Sorted: asdfg
53: Greater: Sorted: 1234567890
54:
```

```
1: ==8150== Memcheck, a memory error detector
2: ==8150== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
3: ==8150== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright in
fo
4: ==8150== Command: insertlist foo bar baz qux quux zxcvbnm asdfg qwerty b
ar baz foo quux qwerty hello hello 1234567890
5: ==8150== Parent PID: 8149
6: ==8150==
7: ==8150==
8: ==8150== HEAP SUMMARY:
9: ==8150==      in use at exit: 0 bytes in 0 blocks
10: ==8150==    total heap usage: 55 allocs, 55 frees, 1,339 bytes allocated
11: ==8150==
12: ==8150== All heap blocks were freed -- no leaks are possible
13: ==8150==
14: ==8150== For counts of detected and suppressed errors, rerun with: -v
15: ==8150== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6)
```