```
 1: // $Id: sortedlist.cpp,v 1.12 2019-02-07 13:54:19-08 - - $
 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 item_t, class less_t=xless<item_t>>
31: struct sorted_list {
32:    struct node {
33:       item_t item;
34:       node* link;
35:       node (const item_t& item_, node* link_):
36:                   item(item_),   link(link_) {
37:       }
38:    };
39:    less_t less;
40:    node* head = nullptr;
41:
42:    sorted_list() = default; // Needed because default is suppressed.
43:    sorted_list (const sorted_list&) = delete;
44:    sorted_list& operator= (const sorted_list&) = delete;
45:    ~sorted_list();
46:
47:    void insert (const item_t& newitem);
48:    item_t& front() { return head->item; }
49:    void pop_front();
50: };
51:
52: template <typename item_t, class less_t>
53: sorted_list<item_t,less_t>::~sorted_list() {
54:    while (head != nullptr) pop_front();
55: }
56:
57: template <typename item_t, class less_t>
58: void sorted_list<item_t,less_t>::insert (const item_t& newitem) {
59:    node** curr = &head;
60:    while (*curr != nullptr and less ((*curr)->item, newitem)) {
61:       curr = &(*curr)->link;
62:    }
63:    if (*curr == nullptr or less (newitem, (*curr)->item)) {
64:       *curr = new node (newitem, *curr);
65:    }
66: }
67:
68: template <typename item_t, class less_t>
69: void sorted_list<item_t,less_t>::pop_front() {
70:    if (head == nullptr) throw underflow_error (__PRETTY_FUNCTION__);
71:    node* old = head;
72:    head = head->link;
73:    delete old;
74: }
```

```
 75:
 76: template <typename item_t, class less_t=xless<item_t>>
 77: void process (int argc, char** argv, const string& label) {
 78:    sorted_list<string,less_t> list;
 79:    for (char** argp = &argv[1]; argp != &argv[argc]; ++argp) {
 80:       cout << label << ": Insert: " << *argp << endl;
 81:       list.insert (*argp);
 82:    }
 83:    cout << endl;
 84:    for (auto itor = list.head; itor != nullptr; itor = itor->link) {
 85:       cout << label << ": Sorted: " << itor->item << endl;
 86:    }
 87:    cout << endl;
 88: }
 89:
 90: int main (int argc, char** argv) {
 91:    process<string> (argc, argv, "Default");
 92:    process<string,xgreater<string>> (argc, argv, "Greater");
 93:    return 0;
 94: }
 95:
 96: /*
 97: //TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
 98: //TEST// grind --log-file=sortedlist.out.log \
 99: //TEST//       sortedlist foo bar baz qux zxcvbnm asdfg qwerty \
100: //TEST//       bar baz foo qwerty hello hello 01234 56789 \
101: //TEST//       >sortedlist.out 2>&1
102: //TEST// mkpspdf sortedlist.ps sortedlist.cpp* sortedlist.out*
103: */
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting sortedlist.cpp
    2: checksource sortedlist.cpp
    3: ident sortedlist.cpp
    4: sortedlist.cpp:
    5:      $Id: sortedlist.cpp,v 1.12 2019-02-07 13:54:19-08 - - $
    6: cpplint.py.perl sortedlist.cpp
    7: Done processing sortedlist.cpp
    8: g++ -Wall -Wextra -Wpedantic -Wshadow -fdiagnostics-color=never -std=gnu
++2a -Wold-style-cast -g -O0 sortedlist.cpp -o sortedlist -lm
    9: rm -f sortedlist.o
   10: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished sortedlist.cpp
```

```
 1: Default: Insert: foo
 2: Default: Insert: bar
 3: Default: Insert: baz
 4: Default: Insert: qux
 5: Default: Insert: zxcvbnm
 6: Default: Insert: asdfg
 7: Default: Insert: qwerty
 8: Default: Insert: bar
 9: Default: Insert: baz
10: Default: Insert: foo
11: Default: Insert: qwerty
12: Default: Insert: hello
13: Default: Insert: hello
14: Default: Insert: 01234
15: Default: Insert: 56789
16:
17: Default: Sorted: 01234
18: Default: Sorted: 56789
19: Default: Sorted: asdfg
20: Default: Sorted: bar
21: Default: Sorted: baz
22: Default: Sorted: foo
23: Default: Sorted: hello
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: zxcvbnm
33: Greater: Insert: asdfg
34: Greater: Insert: qwerty
35: Greater: Insert: bar
36: Greater: Insert: baz
37: Greater: Insert: foo
38: Greater: Insert: qwerty
39: Greater: Insert: hello
40: Greater: Insert: hello
41: Greater: Insert: 01234
42: Greater: Insert: 56789
43:
44: Greater: Sorted: zxcvbnm
45: Greater: Sorted: qwerty
46: Greater: Sorted: qux
47: Greater: Sorted: hello
48: Greater: Sorted: foo
49: Greater: Sorted: baz
50: Greater: Sorted: bar
51: Greater: Sorted: asdfg
52: Greater: Sorted: 56789
53: Greater: Sorted: 01234
54:
```

```
    1: ==25425== Memcheck, a memory error detector
    2: ==25425== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al
.
    3: ==25425== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright
info
    4: ==25425== Command: sortedlist foo bar baz qux zxcvbnm asdfg qwerty bar b
az foo qwerty hello hello 01234 56789
    5: ==25425== Parent PID: 25424
    6: ==25425==
    7: ==25425==
    8: ==25425== HEAP SUMMARY:
    9: ==25425==     in use at exit: 0 bytes in 0 blocks
   10: ==25425==   total heap usage: 52 allocs, 52 frees, 1,264 bytes allocated
   11: ==25425==
   12: ==25425== All heap blocks were freed -- no leaks are possible
   13: ==25425==
   14: ==25425== For counts of detected and suppressed errors, rerun with: -v
   15: ==25425== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```