19 Custom Templatized Data Structures

Objectives

In this chapter you'll:

- Form linked data structures using pointers, self-referential classes and recursion.
- Create and manipulate dynamic data structures such as linked lists, queues, stacks and binary trees.
- Use binary search trees for high-speed searching and sorting.
- Learn important applications of linked data structures.
- Create reusable data structures with class templates, inheritance and composition.
- Have the opportunity to try many challenging data-structures exercises, including the Building Your Own Compiler project.

Self-Review Exercises

19.1 Fill in the blanks in each of the following:

a) A self-_____ class is used to form dynamic data structures that can grow and shrink at execution time

ANS: referential.

b) The _____ operator is used to dynamically allocate memory and construct an object; this operator returns a pointer to the object.

ANS: `new`.

c) A(n) _____ is a constrained version of a linked list in which nodes can be inserted and deleted only from the start of the list and node values are returned in last-in, first-out order.

ANS: stack.

d) A function that does not alter a linked list, but looks at the list to determine whether it's empty, is an example of a(n) _____ function.

ANS: predicate.

e) A queue is referred to as a(n) _____ data structure, because the first nodes inserted are the first nodes removed.

ANS: first-in, first-out (FIFO).

f) The pointer to the next node in a linked list is referred to as a(n) _____.

ANS: link.

g) The _____ operator is used to destroy an object and release dynamically allocated memory.

ANS: `delete`.

h) A(n) _____ is a constrained version of a linked list in which nodes can be inserted only at the end of the list and deleted only from the start of the list.

ANS: queue.

i) A(n) _____ is a nonlinear, two-dimensional data structure that contains nodes with two or more links.

ANS: tree.

j) A stack is referred to as a(n) _____ data structure, because the last node inserted is the first node removed.

ANS: last-in, first-out (LIFO).

k) The nodes of a(n) _____ tree contain two link members.

ANS: binary.

l) The first node of a tree is the _____ node.

ANS: root.

m) Each link in a tree node points to a(n) _____ or _____ of that node.

ANS: child or subtree.

n) A tree node that has no children is called a(n) _____ node.

ANS: leaf.

o) The four traversal algorithms we mentioned in the text for binary search trees are _____, _____, _____ and _____.

ANS: inorder, preorder, postorder and level order.

19.2 What are the differences between a linked list and a stack?

ANS: It's possible to insert a node anywhere in a linked list and remove a node from anywhere in a linked list. Nodes in a stack may only be inserted at the top of the stack and removed from the top of a stack.

19.3 What are the differences between a stack and a queue?

ANS: A queue data structure allows nodes to be removed only from the head of the queue and inserted only at the tail of the queue. A queue is referred to as a first-in, first-out (FIFO) data structure. A stack data structure allows nodes to be added to the stack and removed from the stack only at the top. A stack is referred to as a last-in, first-out (LIFO) data structure.

19.4 Perhaps a more appropriate title for this chapter would have been "Reusable Data Structures." Comment on how each of the following entities or concepts contributes to the reusability of data structures:

a) classes

ANS: Classes allow us to instantiate as many data structure objects of a certain type (i.e., class) as we wish.

b) class templates

ANS: Class templates enable us to instantiate related classes, each based on different type parameters—we can then generate as many objects of each template class as we like.

c) inheritance

ANS: Inheritance enables us to reuse code from a base class in a derived class, so that the derived-class data structure is also a base-class data structure (with `public` inheritance, that is).

d) `private` inheritance

ANS: Private inheritance enables us to reuse portions of the code from a base class to form a derived-class data structure; because the inheritance is `private`, all `public` base-class member functions become `private` in the derived class. This enables us to prevent clients of the derived-class data structure from accessing base-class member functions that do not apply to the derived class.

e) composition

ANS: Composition enables us to reuse code by making a class object data structure a member of a composed class; if we make the class object a `private` member of the composed class, then the class object's `public` member functions are not available through the composed object's interface.

19.5 Provide the inorder, preorder and postorder traversals of the binary search tree of Fig. 19.1.

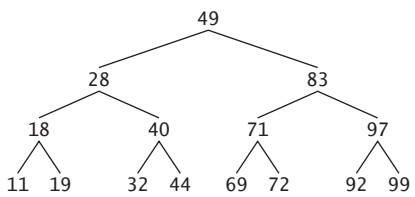

Fig. 19.1 | A 15-node binary search tree.

ANS: The inorder traversal is

```
11 18 19 28 32 40 44 49 69 71 72 83 92 97 99
```

The preorder traversal is

```
49 28 18 11 19 40 32 44 83 71 69 72 97 92 99
```

The postorder traversal is

```
11 19 18 32 44 40 28 69 72 71 92 99 97 83 49
```

Exercises

NOTE: Solutions to the programming exercises are located in the `ch19solutions` *folder.*