float * p_fees;  // = new float[20] not allowed here
int main()
{
    p_fees = new float [20];
...

Compatibility Note

Memory allocated by new typically is freed when the program terminates. However, this is not always true. Under DOS, for example, in some circumstances a request for a large block of memory can result in a block that is not deleted automatically when the program terminates.

Namespaces

Names in C++ can refer to variables, functions, structures, enumerations, classes, and class and structure members. When programming projects grow large, the potential for name conflicts increases. When you use class libraries from more than one source, you can get name conflicts. For example, two libraries might both define classes named List, Tree, and Node, but in incompatible ways. You might want the List class from one library and the Tree from the other, and each might expect its own version of Node. Such conflicts are termed namespace problems.

The C++ Standard provides namespace facilities to provide greater control over the scope of names. It has taken a while for compilers to incorporate namespaces, but, by now, support has become common.

Traditional C++ Namespaces

Before looking at the new facilities, let's review the namespace properties that already exist in C++ and introduce some terminology. This can help make the idea of namespaces seem more familiar.

The first term is declarative region. A declarative region is a region in which declarations can be made. For example, you can declare a global variable outside of any function. The