// explicit specialization for the job type
template <> void Swap<job>(job &, job &);
int main()
{
    double u, v;
    ...
    Swap(u,v); // use template
    job a, b;
    ...
    Swap(a,b); // use void Swap<job>(job &, job &)
}

The <job> in Swap<job> is optional, for the function argument types indicate that this is a specialization for job. Thus, the prototype also can be written this way:

template <> void Swap(job &, job &); // simpler form

The template function heading also can be simplified by omitting the <int> part.

In case you have to work with an older compiler, we'll come back to pre-Standard usage soon, but first, let's see how explicit specializations are supposed to work.

**An Example**

**Listing 8.11** illustrates how explicit specialization works. It's set up to follow the C++ Standard.

**Listing 8.11 twoswap.cpp**

// twoswap.cpp -- specialization overrides a template
#include <iostream>
using namespace std;
template <class Any>
void Swap(Any &a, Any &b);

struct job