If you remove template B from the program, the compiler will then use template A for listing the contents of `pd`, so it will list the addresses instead of the values.

In short, the overload resolution process looks for a function that's the best match. If there's just one, that function is chosen. If there are more than one otherwise tied, but only one is a non-template function, that's chosen. If there are more than one candidates otherwise tied, and all are template functions, but one template is more specialized than the rest, that is chosen. If there are two or more equally good non-template functions, or if there are two or more equally good template functions, none of which is more specialized than the rest, the function call is ambiguous and an error. If there are no matching calls, of course, that also is an error.

**Functions with Multiple Arguments**

Where matters really get involved is when a function call with multiple arguments is matched to prototypes with multiple arguments. The compiler must look at matches for all the arguments. If it can find a function that is better than all the other viable functions, it's the chosen one. For one function to be better than another function, it has to provide at least as good a match for all arguments and a better match for at least one argument.

This book does not intend to challenge the matching process with complex examples. The rules are there so that there is a well-defined result for any possible set of function prototypes and templates.

**Summary**

C++ has expanded C function capabilities. By using the inline keyword with a function definition and by placing that definition ahead of the first call to that function, you suggest to the C++ compiler that it make the function inline. That is, instead of having the program jump to a separate section of code to execute the function, the compiler replaces the function call with the corresponding code inline. An inline facility should be used only when the function is short.

A reference variable is a kind of disguised pointer that lets you create an alias (a second name) for a variable. Reference variables primarily are used as arguments to functions processing structures and class objects.