function call \texttt{Swap(i,j)} causes the compiler to generate an instantiation of the \texttt{Swap()} using \texttt{int} as the type. The template \textit{is not} a function definition, but the specific instantiation using \texttt{int} \textit{is} a function definition. This type of instantiation is termed an \textit{implicit instantiation}, because the compiler deduces the necessity for making the definition by noting that the program uses a \texttt{Swap()} function with \texttt{int} parameters.

Originally, implicit instantiation was the only way the compiler generated function definitions from templates, but now C++ allows for \textit{explicit instantiation}. That means you can instruct the compiler to create a particular instantiation, for example, \texttt{Swap<int>()}, directly. The syntax is to declare the particular variety you want, using the \texttt{<>} notation to indicate the type and prefixing the declaration with the keyword \texttt{template}:

\begin{verbatim}
template void Swap<int>(int, int); // explicit instantiation
\end{verbatim}

A compiler that implements this feature will, upon seeing this declaration, use the \texttt{Swap()} template to generate an instantiation using the \texttt{int} type. That is, this declaration means "Use the \texttt{Swap()} template to generate a function definition for the \texttt{int} type."

Contrast the explicit instantiation with the explicit specialization, which uses one or the other of these equivalent declarations:

\begin{verbatim}
template <> void Swap<int>(int &, int &); // explicit specialization
template <> void Swap(int &, int &); // explicit specialization
\end{verbatim}

The difference is that these declarations mean "Don't use the \texttt{Swap()} template to generate a function definition. Instead, use a separate, specialized function definition explicitly defined for the \texttt{int} type." These prototypes have to be coupled with their own function definitions.

\textbf{Caution}

\begin{itemize}
  \item It is an error to try to use both an explicit instantiation and an explicit specialization for the same type(s) in the same programming unit.
\end{itemize}

Implicit instantiations, explicit instantiations, and explicit specializations collectively are termed \textit{specializations}. What they all have in common is that they represent a function