Generic Functions

• Many operations, when performed on data of different types, are still essentially the same
• Writing different functions for different types is cumbersome, even with overloading
• C++ allows generic functions to alleviate this difficulty
Function Templates

- Form:
  \texttt{template<typename T> func-def}
- \texttt{T} is a \textit{generic type} that can be used anywhere within \texttt{func-def}
- Can use more than one generic type, in comma-separated list between \texttt{<>}'s (e.g. \texttt{<typename T_1, typename T_2>})
- To call function: \texttt{func-name(func-args)}, or \texttt{func-name<T>(func-args)} in case of name conflict with regular function

Overloading

- Can overload function templates with regular functions
- Can overload templates with other templates, as long as return types are consistent
- Error if compiler cannot \textit{unambiguously} determine which overload to use
- Caution: with function templates, compiler may not perform automatic conversions (such as \texttt{double to int})
Specializing Templates

• Might want to make function template behave differently for particular types
• To specialize, leave <>'s blank:
  \texttt{template<> func-def}
• Return type and argument types of \texttt{func-def} are the result of substituting specific types for the \texttt{typename} parameters in the original template
• Will be selected by the compiler instead of the original function template

Instrumenting Code

• When a function or function template is overloaded, argument conversions and name conflicts can make it difficult to determine which definition is used
• Can understand program behavior by \textit{instrumenting} code: adding statements that output useful information about what the code is doing
• Very useful in diagnosing the most frustrating bugs in your code
Template Classes

• A template class is a class that, like a function template, accepts types as parameters
• Examples: `std::vector<int>`, `std::map<std::string, int>`, other container classes
• Can define your own template classes
• Must include implementation in header file, or compiler can’t check operations!

Next Time

All about exception handling