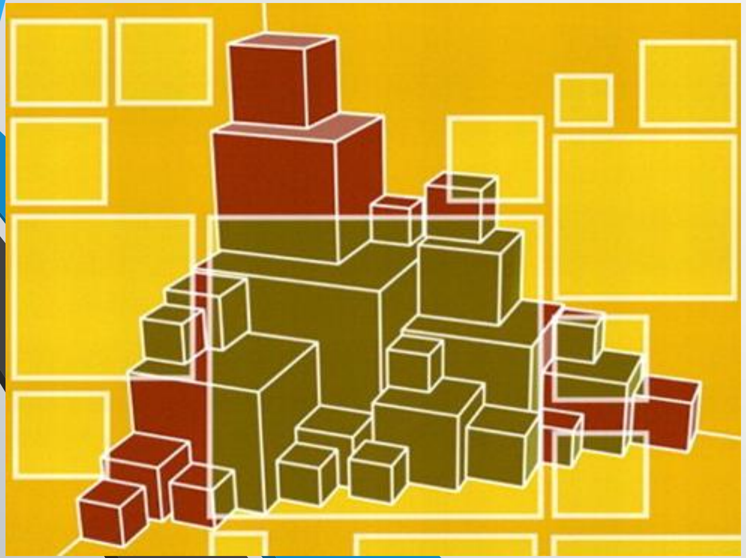

Advanced C++ Programming

Generic Programming with Templates

Preliminaries

Overview & Goals

- To sustainably build large-scale software, we
  - Want to maximize **code re-use**
  - Which means building general foundations that can be specialized for specific use cases
- In C++, as always, we want to achieve this *without runtime overhead*
- **Templates** are the answer

Categories of Templates

- Major:
  - **Function** templates
    Allow the specification of a generic family of functions.
  - **Class** templates
    Allow the specification of a generic family of types.
- Minor:
  - **Alias** templates
    Provide an alias to a family of types
  - **Variable** templates
    Allow the specification of a family of variables

Function Templates

Basic Function Templates

- We'll look at a first example in 04_01_function_templates_basic.cpp
- `template<typename T>` introduces a *template type parameter* T
- Some concrete type will be substituted for this parameter at every call site
- Binary result is the same as if you had implemented each generated function manually

Template Parameters & Arguments

- Just like for functions: *parameters* at declaration site, *arguments* at call site

Template *Parameter* Categories
Non-type template parameters
Type template parameters
Template template parameters

04_02_template_parameter_categories.cpp

+ Parameter Packs of each category

04_03_template_parameter_packs.cpp

Template *Arguments* either
Explicitly specified;
Deduced from the context; or
Defaulted at the declaration

04_04_template_arguments.cpp

Class Templates

Class Template Basics

- Template *Parameters* of the same categories and structure as for function templates
- Basic example in 04_05_class_template_basics.cpp
- Just like for functions, the semantics and resulting code are the same as if you had manually implemented each instantiation of the class

(Partial) Template Specialization

- Templates can be **partially** or **fully** specialized
- Specialization provides *a specific code/data structure version* for cases where some template parameters are bound to specific instances
- We can see an example of this in 04_06_template_specialization.cpp

Class Template Argument Deduction

- Two options for class template *argument deduction* (since C++17)
  - based on the constructor,
  - or manually provided **deduction guides**
    → See 04_07_class_template_arg_deduction.cpp

- Full details:
  http://en.cppreference.com/w/cpp/language/class_template_argument_deduction

Other Templates

Variable Templates

- Not a common use case, primarily for constants
- Basic example in 04_08_variable_templates.cpp
- Common purposes:
  - Replace workarounds such as `constexpr static` members of class templates
    - E.g. for `numeric_limits`
  - Allow you to use constants of the correct type in function and class templates

Alias Templates

- Standard aliases are introduced with “using”
- Alias templates simply apply template syntax to using
- Example in 04_09_alias_templates.cpp

Two-phase Name Lookup

Two Phases of Template Parsing

- Template parsing occurs in two phases:
  1. When encountering the template itself
  2. Whenever it is instantiated
- Identifiers within the template belong in one of two classes:
  - **Non-dependent** names, which are resolved during Phase 1
  - **Dependent** names, which are resolved during Phase 2

Example in **04_10_two_phase_lookup.cpp**

Parsing Ambiguities

- With dependent names referring to templates, parsing can be ambiguous
  - Does "`T::foo<5`" check whether the variable foo is less than 5?
  - Or is it a start to a template function call to the function template "`foo`"?
- Example in 04_11_parsing_hints.cpp
- We need to manually disambiguate by writing

```
T::template foo<5>()
```

Conclusion

Summary

- Templates allow specifying *generic* data structures and algorithms
  - Categories: Function templates, Class templates, Variable templates, Alias templates
- *Instantiation* (for specific types/constants) occurs at **compile time**
- Two-phase parsing can lead to tricky name lookup results