Modern C++ Camp

现代C++系统研发骨干特训营

李建忠 Boolan

Boolan
高端IT咨询与教育平台

现代C++系统研发骨干特训营

模块四、模板机制

模板与泛型编程

- 面向对象（编程范式）
  - 封装
  - 继承
  - 多态
  - 运行时抽象
- 泛型编程（编程范式）
  - 类模板
  - 函数模板
  - 概念
  - 编译时抽象

泛型编程应该成为正常编程活动的一部分

Bjarne Stroustrup

C++模板简介

- C++模板是一种编译时机制，在编译时生成具体的代码，使用实参将模板定义实例化为具体的类型或函数。C++支持两种模板：
  - 类模板
  - 函数模板

- 模板实例化时编译器会对实参类型进行检查，确保实参符合对模板参数的操作要求。 C++模板参数支持两种：
  - 类型参数，可隐式约束、也可显式约束
  - 值参数，编译时常量、或constexpr函数。不同值参数是不同类型。
  - 可为模板参数提供默认值

- 对模板参数进行显式约束，即C++ 20的概念（Concept）

模板类成员

- 普通成员：使用与主模板相同类型模板参数
  - 数据成员（变量、常量）
  - 成员函数
  - 静态成员（数据或函数）
  - 成员类型别名
  - 成员类型
- 成员模板（使用与主模板不同的类型参数）
  - 成员模板不能定义虚函数（模板实例化会导致链接器不断为虚表增加虚函数增项）
- 所有普通类的成员规则同样适用于模板类成员

模板实例化机制

- 数据成员——只要类型被使用，编译器会根据其数据成员、生成对应类型结构。
- 函数成员——选择性实例化
  - 非虚函数，如果实际调用到，则会生成代码；如果没有调用到，则不生成。
  - 虚函数，无论是否调用，总会生成代码。因为在运行时“有可能”调用到。
- 隐藏编译错误
  - 如果某些模板方法没有被调用，即使包含编译错误，也会被忽略。
- 强制实例化模板
  - 使用template class Array<int>; 来强制要求编译所有模板类函数成员，排除所有编译错误，无论是否调用到。

类型别名与模板别名

- 为模板类使用指定别名
  - 类型别名（alias type）：指定所有模板参数，得到完整类型
  - 模板别名（alias template）：指定部分参数，得到模板类型
  - 成员类型别名：类模板中通过定义类型别名，来定义“关联类型”
  - 别名和原始模板完全等价，包括使用模板特化时（但不支持特化别名）

- 优先使用using 而不是typedef
  - 两者都可以声明类型别名、成员类型别名
  - using 可以定义别名模板，而typedef不可以
  - using 可以免掉类型内typedef要求的typename前缀，和::type 后缀

模板参数类型自动推导

- C++模板编译时支持对类型参数的自动推导：
  - 普通函数参数
  - 类成员函数参数
  - 构造函数参数（C++ 17 支持），模板所有类型参数都有值

- 模板类型推导时：
  - 引用性被忽略：引用类型实参被当作非引用类型处理。
  - 转发引用时，左值参数按左值、右值参数按右值。
  - 按值传递时，实参中const/volatile修饰会被去掉。

模板特化

- 模板类型的特化指的是允许模板类型定义者为模板根据实参的不同，而定义不同实现的能力。
- 特化版本可以和一般版本拥有不同的外部接口和实现。
- 模板偏特化：也可以对部分模板参数进行特化定义，称为偏特化。
- 模板特化支持模板类、模板函数。

奇异递归模板模式

Curiously Recurring Template Pattern，简称CRTP，通过将基类模板参数设置为子类，从而实现静态多态（静态接口），或者扩展接口（委托实现）。

CRTP实现要点解析：

- class Sub : public Base<Sub> 通过模板参数，将子类类型在编译时注入基类，从而实现在基类中提前获取子类类型信息。
- static_cast<T*>(this) 将基类指针转型为模板子类T的指针
- Base类型为不完整类型，不能使用Sub参与内存布局，但可以在函数内使用（发生调用，模板编译时辨析即可）
- 删除对象，也要使用编译时多态进行删除，避免直接delete