深蓝学院
shenlanxueyuan.com

C++ 基础

第 14 章：元编程

主讲人 李伟

微软高级工程师

《C++ 模板元编程实战》作者

目录

元编程的引入

- 从泛型编程到元编程
  - 泛型编程——使用一套代码处理不同类型
  - 对于一些特殊的类型需要引入额外的处理逻辑——引入操纵程序的程序
  - 元编程与编译期计算
- 第一个元程序示例（Erwin Unruh）
  - 在编译错误中产生质数
- 使用编译期运算辅助运行期计算
  - 不是简单地将整个运算一分为二
  - 详细分析哪些内容可以放到编译期，哪些需要放到运行期
    - 如果某种信息需要在运行期确定，那么通常无法利用编译期计算

元编程的引入——续

- 元程序的形式
  - 模板， constexpr 函数，其它编译期可使用的函数（如 sizeof ）
  - 通常以函数为单位，也被称为函数式编程
- 元数据
  - 基本元数据：数值、类型、模板
  - 数组
- 元程序的性质
  - 输入输出均为“常量”
  - 函数无副作用
- type_traits元编程库
  - C++11 引入到标准中，用于元编程的基本组件

顺序、分支、循环代码的编写方式

- 顺序代码的编写方式
  - 类型转换示例：为输入类型去掉引用并添加 const
  - 代码无需至于函数中
    - 通常置于模板中，以头文件的形式提供
  - 更复杂的示例：
    - 以数值、类型、模板作为输入
    - 以数值、类型、模板作为输出
  - 引入限定符防止误用
  - 通过别名模板简化调用方式

顺序、分支、循环代码的编写方式

- 分支代码的编写方式
  - 基于 if constexpr 的分支：便于理解只能处理数值，同时要小心引入运行期计算
  - 基于（偏）特化引入分支：常见分支引入方式但书写麻烦
  - 基于 std::conditional 引入分支：语法简单但应用场景受限
  - 基于 SFINAE 引入分支
    - 基于 std::enable_if 引入分支：语法不易懂但功能强大
      - 注意用做缺省模板实参不能引入分支！
    - 基于 std::void_t 引入分支： C++17 中的新方法，通过“无效语句”触发分支
  - 基于 concept 引入分支： C++20 中的方法
    - 可用于替换 enable_if
  - 基于三元运算符引入分支： std::conditional 的数值版本

顺序、分支、循环代码的编写方式

- 循环代码的编写方式
  - 简单示例：计算二进制中包含 1 的个数
  - 使用递归实现循环
  - 任何一种分支代码的编写方式都对应相应的循环代码编写方式
  - 使用循环处理数组：获取数组中 id=0,2,4,6... 的元素
  - 相对复杂的示例：获取数组中最后三个元素

减少实例化技巧

- 为什么要减少实例化
  - 提升编译速度，减少编译所需内存
- 相关技巧
  - 提取重复逻辑以减少实例个数
  - conditional 使用时避免实例化
  - 使用 std::conjunction / std::disjunction 引入短路逻辑
- 其它技巧介绍
  - 减少分摊复杂度的数组元素访问操作

深蓝学院
shenlanxueyuan.com

感谢聆听！

Thanks for Listening