第36章 typename优化

《现代C++语言核心特性解析》 谢丙堃

允许使用typename声明模板模板形参

- 在C++17之前只能使用class声明模板模板形参

```
template <typename T> struct A {};
template <template <typename> class T> struct B {};
int main()
{
    B<A> ba;
}
```

```
template <typename T> struct A {};
template <template <typename> typename T> struct B {};
int main()
{
    B<A> ba;
}
```

允许使用typename声明模板模板形参

- 从C++11开始，模板模板形参并不一定只有类模板

```
template <typename T> using A = int;
template <template <typename> typename T> struct B {};
int main()
{
    B<A> ba;
}
```

减少typename使用的必要性

- 对于未决类型的内嵌类型需要用到typename

```
template<class T> void f(T::R); // 编译错误

template<class T> void f(typename T::R);
```

减少typename使用的必要性

- 在C++20标准之前，只有两种情况例外，它们分别是指定基类和成员初始化：

```
struct Impl {};

struct Wrap {
    using B = Impl;
};

template<class T>
struct D : T::B {
    D() : T::B() {}
};

D<Wrap> var;
```

减少typename使用的必要性

- C++20标准增加省略typename关键字场景：
  1. 在上下文仅可能是类型标识
     - 比如 static_cast、 const_cast、 reinterpret_cast或者dynamic_cast等类型转换
     - 或者定义类型别名
     - 或者后置返回类型
     - 还有比如模板类型形参的默认参数
  2. 其他可以忽略typename的场景
     - 比如在全局或者命名空间中简单的声明或者函数的定义
     - 又或者是结构体的成员类型
     - 另外就是当其作为成员函数或者lambda表达式形参声明的时候

感谢您的观看

欢迎关注