第42章 模板特性的其他优化

《现代C++语言核心特性解析》 谢丙堃

外部模板

- 模板的重复实例化问题:

```
// header.h
template<class T> bool foo(T t) { return true; }

// src1.cpp
#include <header.h>
bool b = foo(7);

// src2.cpp
#include <header.h>
bool b = foo(11);
```

外部模板

- 使用extern解决模版重复实例化：

```
// header.h
template<class T> bool foo(T t) { return true; }

// src1.cpp
#include <header.h>
extern template bool foo<double>(double);

// src2.cpp
#include <header.h>
template bool foo<double>(double);
```

连续右尖括号的解析优化

- C++11之前连续的右尖括号会被识别为右移：

```
#include <vector>
typedef std::vector<std::vector<int> > Table; // 编译成功
typedef std::vector<std::vector<bool>> Flags; // 编译失败，>>被解析为右移
```

连续右尖括号的解析优化

- C++11支持连续右尖括号后的兼容问题：

```
template<int N>
class X {};

X <1 >> 3> x;
```

friend声明模板形参

- 忽略class的友元声明：

```
class C;

class X1 {
    friend class C; // C++11前后都能编译成功
};

class X2 {
    friend C; // C++11以前会编译错误，C++11以后编译成功
};
```

friend声明模板形参

- 比如用friend声明基本类型，用friend声明别名，用friend声明模板参数：

```
class C;
typedef C Ct;

class X1 {
    friend C;
    friend Ct;
    friend void;
    friend int;
};

template <typename T> class R {
    friend T;
};
```

变量模版

- 从C++14开始通过模板不仅能实例化类和函数，还可以实例化变量了：

```
template<class T>
constexpr T PI = static_cast<T>(3.14159265358979323851L);
int main()
{
    std::cout << PI<float> << std::endl;
}
```

变量模版

- 不必一定是常量，变量也可能定义：

```
template<class T, int N>
T PI = static_cast<T>(3.14159265358979323851L) * N;
int main()
{
    PI<float, 2> *= 5;
    std::cout << PI<float, 2> << std::endl;
}
```

explicit(bool)

- C++20开始可以接受一个求值类型为bool的常量表达式，用于指定explicit的功能是否生效：
  - 因为有时候需要根据参数具体类型来决定是否支持隐式调用。

感谢您的观看

欢迎关注