**C++学习笔记——type\_traits**

C++的头文件<type\_traits>中包含了很多与模板类型相关的操作，多以模板结构体形式出现。本文记录一些常见的用法。

**1. std::enable\_if<T>**

图形用户界面, 文本, 应用程序

描述已自动生成

enable\_if是type\_traits中的一个模板结构体，其包含两个模板参数：第一个是bool，第二个是自定义类型。

如果条件\_Cond为true，std::enable\_if将被偏特化为拥有一个公共成员type，等价于T；否则，没有成员typedef，即此模板不存在。

可以使用std::enable\_if对函数模板进行条件限制。当某个条件为真时，std::enable\_if可以使函数模板“启用”，否则“禁用”函数模板。这个功能可以使得函数模板更加通用、可靠和安全。

**2. std::remove\_reference<T>**

文本

描述已自动生成

std::remove\_reference是type\_traits中的一个模板结构体。

如果传入的模板参数\_Tp是引用类型（包括左值和右值引用），则返回引用所指向的类型，相当于“移除引用”；否则返回该类型本身。

要想获取移除引用后的类型，须使用：

std::remove\_reference<T>::type，或std::remove\_reference\_t<T>。

**3. std::decay<T>**

移除const、引用限定，返回原生类型。

如果T是const或引用类型，则移除const和引用；

如果T是数组类型，则返回指向数组元素类型的指针；

如果T是函数类型，则返回函数指针；

如果T没有以上任何限定，则返回T本身。

要想获取decay之后的类型，须使用std::decay<T>::type，或std::decay\_t<T>。

例如：

std:decay\_t<int&> // int

std::decay\_t<int&&> // int

std::decay\_t<const double&> // double

**4. std::is\_lvalue\_reference<T> / std::is\_rvalue\_reference<T>**

判断一个引用类型是否为左值引用/右值引用。

要想获取判断结果，须使用：

std::is\_lvalue\_reference<T>::value / std::is\_rvalue\_reference<T>::value

或：

std::is\_lvalue\_reference\_v<T> / std::is\_rvalue\_reference\_v<T>