|  |  |
| --- | --- |
| 1. 转发 2. 定义能保持类型信息的函数参数 3. 在调用中使用std::forward保持类型信息 | Template<typename F, typename T1, typename T2>  Void flip1(F f, T1 t1, T2 t2)  {  F(t2,t1);  }  Void f(int v1,int &v2){}  Flip1(f,j,42);  Template<typename F,typename T1,typename T2>  Void flip2(F f,T1 &&t1,T2 &&t2)  {  f(t2,t1) ;  }  Flip2(f,j,42);  Void g(int &&i,int& j){}  Flip2(g,I,42);  Template <typename type> intermediary(Type &&arg)  {  finalFcn(std::forward<Type>(arg));  } |
| 1. 转发 2. 当将实参连同类型不变地转发给其它函数。我们需要保持被转发实参的所有性质，包括实参类型是否是const的以及实参是左值还是右值 3. 第一段代码，T1的类型为int，所有t1是拷贝，传给f的v2则是对t1的引用，不是对传递实参的j引用 4. 定义能保持类型信息的函数参数 5. 需要将模板类型参数为右值引用，就能保持因为引用折叠，可传左值或右值 6. 第二段代码：T1的类型是int&,t1为j的引用，所以传给f的v2是对j的引用   Note：如果一个函数模板的参数是T&&，对应的实参的const属性和左值/右值属性将得到保持。   1. 对于接受左值引用再给一个函数的左值引用参数工作良好，传j给g函数额j   但是要想接受右值引用再传给一个函数的右值引用参数就不行,传42给 g函数的i  第三段代码：  T2的类型是int，int&& t2,则t2是一个右值引用，但是函数参数与其它任何变量一样，都是左值表达式所以传递给g的第一个参数是将一个左值引用绑定到一个右值引用。因为g函数不是模板所以不能折叠。   1. 在调用中使用std::forward保持类型信息 2. 用forward的新标准设施解决，保持原始原始参数的类型。 3. 需要将函数模板的参数为T&& 4. 记住：Forward<Type>，是将Type为Type&& 5. 当传递右值引用时，T为类型int，那么forward后，则为int&& 6. 当传递左值引用时，因为折叠T为左值引用int&,那么forward时：int& &&,再次折叠又为左值引用   所以，当传给函数模板为右值引用后再传给函数g的右值引用参数是不变的。相对左值引用也不变。  问题：  T&&时得尝试一下是否const左值引用会保持const，会  T为 const int&,函数为const int&   1. Move是为了将左值引用和右值引用接受后还是返回右值引用。 2. Forward是为了将左值引用和右值引用接受后传递给其它不是函数模板的函数还是原来的类型   Note::move forward不使用using声明好主意，p706  std::forward<Type> 给type加上&&，若type为&则 && &折叠为&，若type为int，则&& 为int&& | |