某些函数需要将其一个或多个实参连同类型不变地转发给其他函数。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。

转发的作用：

例：程序C++11\_test15

// 参数转发

// 第一个模板形参F为可调用表达式

// T1和T2为两个额外形参

template<typename F, typename T1, typename T2>

void Flip1(F f, T1 t1, T2 t2)

{

f(t1, t2);

}

void F(int v1, int& v2)

{

++v2;

}

// 转发不完整的情况

*cout* << "转发不完整" << *endl*;

int i1 = 0;

int j1 = 1;

*cout* << "转发" << *endl*;

Flip1(F, i1, j1);

*cout* << "i = " << i1 << " j = " << j1 << *endl*;

*cout* << "直接调用" << *endl*;

int i2 = 0;

int j2 = 1;

F(i2, j2);

*cout* << "i = " << i2 << " j = " << j2 << *endl*;

输出为：

转发不完整

转发

i = 0 j = 1

直接调用

i = 0 j = 2

从输出可以看出：

Flip1进行了函数的实参转发，但其转发并不完整，对于同样的输入参数，直接调用函数F会改变第2个实参的值，但通过Flip1调用F则不会改变第2个实参的值。原因是Flip1的形参t2的类型是非引用类型，Flip1调用实例化为：

void Flip1(void(\*fcn)(int, int&), int t1, int t2)

实参j1被拷贝到t2中，而非被引用，所以修改t2的值不会影响到j1.

更进一步的转发：

例：程序C++11\_test15

// 更近一步的转发

template<typename F, typename T1, typename T2>

void Flip2(F f, T1&& t1, T2&& t2)

{

f(t1, t2);

}

*cout* << "更近一步转发" << *endl*;

int i3 = 0;

int j3 = 1;

Flip2(F, i3, j3);

*cout* << "i = " << i3 << " j = " << j3 << *endl*;

输出为：

更近一步转发

i = 0 j = 2

从输出可以看出j3的值发生了改变，原因是Flip2的模板形参的推断类型为int&，这样经过类型折叠（参考C++STL模板学习3-右值引用和move.docx）后，t2的类型为int&，所以j3被绑定到了t2，t2发生改变，j3也会改变。

存在的问题：无法用于接收右值引用的函数

例：程序C++11\_test15

void G(int& i, int&& j)

{

}

// Flip2(G, i3, 4); // error

函数G的第2个形参接收右值引用，通过Flip2调用G时，传递给G的是t2，尽管t2的类型为int&&，但他本身是一个左值，所以为报错

完美转发：

// 使用std::forward实现完美转发

template<typename F, typename T1, typename T2>

void Flip3(F f, T1&& t1, T2&& t2)

{

f(std::*forward*<T1>(t1), std::*forward*<T2>(t2));

}

*cout* << "完美转发" << *endl*;

int i4 = 0;

int j4 = 1;

Flip3(F, i4, j4);

*cout* << "i = " << i4 << " j = " << j4 << *endl*;

Flip3(G, i4, 8); // ok

输出为：

完美转发

i = 0 j = 2