在实例化函数模板或类模板时，编译器会根据函数实参或类实参来确定模板实参的类型和值。

当模板有多个类型形参时，实参类型必须与之完全匹配。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 文件名 ：test.cpp

/\* 作者 ：刘珅珅

/\* 日期 ：2011/11/8

/\* 功能描述 ：模板实参的匹配

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// 注意写函数模板和类模板的格式

template <typename T>

int compare(const T &v1, const T &v2);

int main(int argc, char \*argv[])

{

short s1 = 10;

// int s2 = 20;

short s3 = 20;

// compare(s1, s2); // error:实参的类型与形参不完全匹配，不能实例化compare(short, int)，必须是compare(short, short)或compare(int, int)，编译器也不会把short型提升为int型

compare(s1, s3);

return 0;

}

template <typename T>

int compare(const T &v1, const T &v2)

{

if (v1 < v2)

{

return -1;

}

if(v2 < v1)

{

return 1;

}

return 0;

}

如果要匹配不同类型的实参，可以修改为：

template <typename A, typename B>

int compare(const A &v1, const B &v2)

{

if (v1 < v2)

{

return -1;

}

if(v2 < v1)

{

return 1;

}

return 0;

}

类型形参的实参的受限转换

short s1, s2;

int i1, i2;

compare(i1, i2); // 实例化compare(int, int)

compare(s1, s2); // 实例化compare(short, short)

可以看出，当实例化了一个compare(int, int)函数后，编译器不会将short型实参转换为int型，而是会重新实例化一个compare(short, short)函数。

一般而言，模板在实例化的过程中，编译器不会转换实参匹配已有的实例，而是去产生新的实例。编译器只会执行两种转换：

1. const转换：接受const引用或const指针的函数可以分别用非const对象的引用或指针来调用，无须产生新的实例化。如果函数接受非引用类型，形参类型和实参都忽略const，即无论传递const或非const对象给接受非引用类型的函数，都使用相同的实例化。

template <typename T>

T fobj(T, T);

template <typename T>

T fref(const T&, const T&);

string s1("a value");

const string s2("another value");

fobj(s1, s2); // ok:const被忽略

fref(s1, s2); // ok:s1被转换为const类型

1. 数组或函数到指针的转换：如果模板形参不是引用类型，则数组或函数类型的实参应用常规指针转换，数组实参转换成指向其第一个元素的指针，函数实参转换成指向函数类型的指针。

template <typename T>

T fobj(T, T);

template <typename T>

T fref(const T&, const T&);

int a[10], b[10];

fobj(a, b); // ok:实例化fobj(int\*, int\*)

fref(a, b); // error:形参为引用，数组不能转换为指针，而fref函数的形参不是数组的引用，与是否为const无关

例：

#include <iostream>

using namespace std;

template <typename T>

void fref( const T (&a)[2], const T (&b)[2])

{

for (size\_t i = 0; i != 2; ++i)

{

cout << a[i] <<"," << b[i] << endl;

}

}

int main(int argc, char \*argv[])

{

int a[2] = {1, 2};

int b[2] = {3, 4};

fref(a, b); //

return 0;

}

如果类模板中有非类型形参时，在实例化时，必须为每个非类型形参提供常量表达式以供使用。

例：

template <int hi, int wid>

class Screen

{

//

};

Screen <24, 80> hp261;