**http://www.cnblogs.com/graphics/archive/2010/07/15/1777760.html**

**[如何使用数组参数](http://www.cnblogs.com/graphics/archive/2010/07/15/1777760.html)**

如果一个函数以一维数组为参数，我们可以这样声明这个函数

void func(int\* a) ;

void func(int a[]) ;

void func(int a[3]) ;

实际上，这三种形式是等价的，在使用数组做参数时，编译器会自动将数组名转换为指向数组第一个元素的指针，为什么呢？这要从参数的传递方式说起，参数有三种传递方式，按值传递，按指针传递，按引用传递，分别如下

void Test(int a) ;

void Test(int\* a) ;

void Test(int& a) ;

第一种方式传递的是a的一个副本

第二种方式传递的是指向a的指针的一个副本

第三种方式传递的是指向a的引用的一个副本

既然都是副本,那么就存在拷贝到过程，但是，数组是不能直接拷贝的，也就是不能像下面这样

int a[3] = {1, 2, 3} ;

int b[](a) ; // error

int b[3] ;

b = a ; // error

不能用一个数组初始化另一个数组，也不能将一个数组直接赋值给另外一个数组，如果想复制数组，唯一的办法就是逐个元素复制。

int a[3] = {1, 2, 3} ;

int b[3] ;

for (int i = 0; i < 3; ++i)

{

b[i] = a[i] ;

}

既然数组不能拷贝，那么参数该如何传递呢？于是编译器就将数组名转换成了指向第一个元素的指针，指针是可以拷贝的。但是这也引发了另外一个问题。我们无法只通过数组名得知数组元素的个数。看下面的代码

void Test(int a[3])

{

for (int i = 0; i < 5; ++i)

{

cout << a[i] << endl ;

}

}

明明只传递了三个元素的数组，为什么输出5个元素？前面已经说了，数组被转换成了指向第一个元素的指针，所以上面的代码和下面的相同

void Test(int\* a) //我只知道a是个指针，跟本不知道a指向多少个元素

{

for (int i = 0; i < 5; ++i)

{

cout << a[i] << endl ;

}

}

编译器根本不知奥数组a有多少个元素，它甚至不知道a是数组！如何解决呢，一种办法是再加一个参数，指定元素个数

void Test(int\* a, int n)

{

for (int i = 0; i < n; ++i)

{

cout << a[i] << endl ;

}

}

另外一种办法是传递数组的引用，这才是本文的重点，唉，前面这么多废话：（

void Test(int (&a)[3])

{

for (int i = 0; i < 3; ++i)

{

cout << a[i] << endl ;

}

}

这样写数组a就不会被转换为指针了，而且有了元素个数的信息，调用的时候，也必须传递一个含有3个元素的数组

int a[3] = {1, 2, 3} ;

Test(a) ; // ok

int b[1] = {1} ;

Test(b) ; // error, can not convert parameter a from int[1] to int(&)[3]

And that’s all, Happy coding!

==The End==