const int \*p, int \*const p, const int \*\*p , int \*\*const p, int \*const \*p的区别

被const修饰的符号我们称为常量，更准确的说应该是常变量，例如：

const int a=10;

我们定义了一个常变量整形a，那么对于a的操作有什么特征呢？当然，你只能使用a给别的变量赋值，但是切记不能给a进行赋值，编译器认为a的内存是不可修改的，因此，试图给a赋值的操作都会失败，编译阶段都不会通过！

1. const int \*p和int \*const p的区别

首先要分清楚const修饰的是谁，只有被const修饰的类型是不能被修改的，这里我借用c++标准对于const说明的一句话，const只修饰离它最近的类型符号，那我们来看：

* const int \*p

可以看出const修饰的是int，也就是说\*p所代表的整型值是不能被修改的，试图通过\*p修改指针所指向的内存数据是非法的，看这个例子：

|  |
| --- |
| int a=10;  int b=20;  const int \*p=&a;  \*p=30; //error! \*p所代表的int是常量（也就是a），不能被修改  p=&b; //ok! p并没有被const所修饰,因此可以修改p的值  \*p=20; //error! 试图修改b的值，由于\*p是常量，因此不能修改 |

可以看出，const int\*p指的是，\*p所代表的整形值是常量，不能被修改；而指针p本身是变量，可以被任意赋值。更通俗的描述就是，指针p所指向的内存不能被修改，而指针p本身是可以被修改的！

另外，const int\*p和int const\*p是一样的，因为在int const \*p中，const的左右两边只有左边int为类型(const是修饰类型的)，右边\*p不是类型，因此把const放到int的左右两边情况是一样的！

* int \*const p

可以看出const修饰的是int \*，也就是这个指针变量p是常量，而\*p所指向的int整形值是变量，请看示例:

|  |
| --- |
| int a=10;  int b=20;  int \*const p=&a;  \*p=30; //ok! \*p是变量，可以被任意修改赋值  p=&b; //error！ p是常量，不能被赋值修改，试图修改p，就会出错！  \*p=20; //ok! \*p是变量，可以被任意修改赋值 |

可以看出,int \*const p指的是，指针p本身是常量，不能被修改，而\*p所代表的整形是变量，可以被任意修改！即，p不能被修改，而p所指向的内存是可以被修改的！

1. const int \*\*p , int \*\*const p, int \*const \*p的区别

参照第一点，你能够很快区分const int \*\*p , int \*\*const p的区别。

const int \*\*p 表示\*\*p代表的整形内存放的是常量，不能被修改，而指针p本身可以被任意修改赋值。

int \*\*const p 表示\*\*p代表的整形内存放的是变量，可以被任意修改，而指针p本身是不能被修改的

int \*const \*p是什么意思呢？

由于int \* const \*p是一个二级指针，那么const现在修饰的是int\*，代表\*p的值是不能被修改的，通俗的说，也就是二级指针p所指向的一级指针\*p是不能被修改的,而\*\*p所代表的整形内存可以被任意修改，指针p本身也可以被任意修改！

对于二级指针综合const有一个重要的问题，如下，请看示例代码：

|  |
| --- |
| int a=10;  int \*p=&a;  const int \*\*q=&p;//error! “初始化”: 无法从“int \*\*”转换为“const int \*\*”  也类似于这样的代码：  int \*\*p;  const int \*\*q=p; //error! “初始化”: 无法从“int \*\*”转换为“const int \*\*” |

可以看出,c++不让我们把int\*\*转换为const int\*\*，这样不好吗？我们不是把类型的安全属性提高了吗？为什么反到禁止这样的类型转换呢？

当然c++这样做是出于安全考虑的，以防止我们修改常量对象。为了展现如何会修改常量对象，在这里我们做个假设，假设c++允许我们代码“const int \*\*q=&p”的执行，也就是允许我们把int\*\*转换为const int\*\*，请看实例：

|  |
| --- |
| int a=10;  int \*p=&a;  const int \*\*q=&p;//假设这一行代码c++允许执行  const int b=20;  const int \*pb=&b;  \*q=pb;//重点在这里，由于\*q（\*q就是指针p）是变量，pb指针也是变量，因此把pb赋值给\*q，也就等于执行了p=pb,由于pb指向的是常量b，因此指针p也指向了常量b，但是看看p的定义 int \*p, 没有任何const修饰，你可以通过\*p修改内存的值，但是\*p代表的常量b，是不能被修改的，你试图通过非常量指针p去修改常对象是错误的！编译器不会答应这样的事情！ |

通过上面例子，你应该知道c++为什么不允许我们把int\*\*转换为const int\*\*，因为这样会有让一个普通指针指向常对象的危险。那我们知道了这一点，如何去修改呢，有两种修改方式，我们来看看：

* 可以把指针p定义为const int \*p

|  |
| --- |
| int a=10;  const int \*p=&a;  const int \*\*q=&p;  const int b=20;  const int \*pb=&b;  \*q=pb; // 现在我让\*q指向的常量b，也就是让p指向了常量b，但是，\*p现在是常量，是不能被赋值的，这样就保护了常对象b的值不会被修改 |

* 可以把指针q的定义改为const int \*const \*q

|  |
| --- |
| int a=10;  int \*p=&a;  const int \*const \*q=&p;//第一个const修饰了int,代表\*\*q的值不能被改变；第二个const修饰了int\*,代表\*q不能被改变，也就是q所指向的指针p不能被赋值修改，因此试图做\*q=pb是不被允许的，因此从源头上杜绝了用一个非常量指针指向常量对象的危险！  const int b=20;  const int \*pb=&b; |

希望朋友们能从这篇文章中有所收获！