问题

常量指针和指针常量有什么区别？

const的优点

在C++中，关键字const用来只读一个变量或对象，它有以下几个优点：

1. **便于类型检查**，如函数的函数 func(const int a) 中a的值不允许变，这样**便于保护实参**。
2. 功能**类似于宏定义，方便参数的修改和调整**。如 const int max = 100；
3. 节省空间，如果再定义a = max,b=max...等就不用在为max分配空间了，而用宏定义的话就一直进行宏替换并为变量分配空间
4. 为函数重载提供参考，即可以添加const进行重载。

常量指针和指针常量的区别

首先要区分常量指针和指针常量分别是什么，这里有一种很好的记忆方法：

指针（*）和常量（const）谁在前先读谁 ；* 象征着地址，const象征着内容；谁在前面谁就不允许改变。

```
int a = 1;
int b = 2;
int c = 3;
int const *p1 = &b;      // const在前，p1为常量指针
int *const p2 = &c;      // * 在前，p2为指针常量
//注意：允许将非const对象的地址赋给指向const对象的指针，所以第4行代码是正确的
```

常量指针p1：即指向const对象的指针，指向的地址可以改变，但其指向的内容（即对象的值）不可以改变。

```
//p1可以改变，但不能通过p1修改其指向的对象（即 b）的值；不过，通过其他方式修改b的值是允许的
p1 = &a;     //正确，p1是常量指针，可以指向新的地址(即&a)，即p1本身可以改变
*p1 = a;     //错误，*p1是指针p1指向对象的值，不可以改变，因此不能对*p重新赋值
```

指针常量p2：指针本身是常量，即指向的地址本身不可以改变，但内容（即对象的值）可以改变。

```
p2 = &a;     //错误，p2是指针常量，本身不可以改变，因此将a的地址赋给p2是错误的
*p2 = a;     //正确，p2指向的对象允许改变
```

补充：要分辨是常量指针还是指针常量，可以从右向左来看其定义，具体如下：

①对于 int const *p1=&b，先将*和p1结合，即p1首先是一个指针，然后再左结合const，即常量指针，它指向了const对象，因此我们不能改变 *p1的值。

②对于 int *const p2=&c，现将const和p2结合，即p2首先是一个常量，然后再左结合*，即指针常量，它本身是一个常量，因此我们不能改变p2本身。另外因为p2本身是const，而const必须初始化，因此p2在定义时必须初始化，即不能直接 int *const p2;

参考资料

常量指针和指针常量的区别详解

《C++ Primer》