C++三种野指针

野指针，也就是指向不可用内存区域的指针。**如果对野指针进行操作，将会使程序发生不可预知的错误，甚至可能直接引起崩溃**。

野指针不是NULL指针，是指向“垃圾”内存的指针。人们一般不会错用NULL指针，因为用if语句很容易判断。但是野指针是很危险的，也具有很强的掩蔽性，if语句对它不起作用。

造成野指针的常见原因有三种：

1、指针变量没有被初始化。任何指针变量刚被创建时不会自动成为NULL指针。在Debug模式下，VC++编译器会把未初始化的栈内存上的指针全部填成 0xcccccccc ，当字符串看就是 “烫烫烫烫……”；会把未初始化的堆内存上的指针全部填成 0xcdcdcdcd，当字符串看就是 “屯屯屯屯……”。把未初始化的指针自动初始化为0xcccccccc或0xcdcdcdcd，而不是就让取随机值，那是为了方便我们调试程序，使我们能够一眼就能确定我们使用了未初始化的野指针。在Release模式下，编译器则会将指针赋随机值，它会乱指一气。所以，指针变量在创建时应当被初始化，要么将其设置为NULL，要么让它指向合法的内存：

```
char *p; //此时p为野指针
```

2、指针指向的内存被释放了，而指针本身没有置NULL。对于堆内存操作，我们分配了一些空间（使用malloc函数、calloc函数或new操作符），使用完后释放（使用free函数或delete操作符）。指针指向的内存被释放了，而指针本身没有置NULL。通常会用语句if (p != NULL)进行防错处理。很遗憾，此时if语句起不到防错作用。因为即便p不是NULL指针，它也不指向合法的内存块。所以在指针指向的内存被释放后，应该将指针置为NULL：

```
char *p=new char[10];  //指向堆中分配的内存首地址，p存储在栈区
cin>> p;
delete []p; //p重新变为野指针
```

3 、指针超过了变量的作用范围。即在变量的作用范围之外使用了指向变量地址的指针。这一般发生在将调用函数中的局部变量的地址传出来引起的。这点容易被忽略，虽然代码是很可能可以执行无误，然而却是极其危险的。局部变量的作用范围虽然已经结束，内存已经被释放，然而地址值仍是可用的，不过随时都可能被内存管理分配给其他变量：

```
char *p=new char[10]; //指向堆中分配的内存首地址
cin>> p;
cout<<*(p+10); //可能输出未知数据
```