C/C++中，为了避免野指针（即指针没有指向任何地址）的出现，声明一个指针后，最好马上对其进行初始化。

如果暂时不明确指针指向哪个变量，则可以赋予NULL，如：

int\* p = NULL;

除了NULL之外，C++11新标准引入了nullptr来表示一个空指针。

nullptr 既不是整型类型，也不是指针类型，nullptr 的类型是 std::nullptr\_t，能转换成任意的指针类型。

**为什么建议使用nullptr代替NULL呢？**

这是因为在C++中，NULL是被定义为0的常量，当遇到函数重载时，就会出现问题。

[C++覆盖和重载的区别](https://www.toutiao.com/i6838597248499057165/?group_id=6838597248499057165)

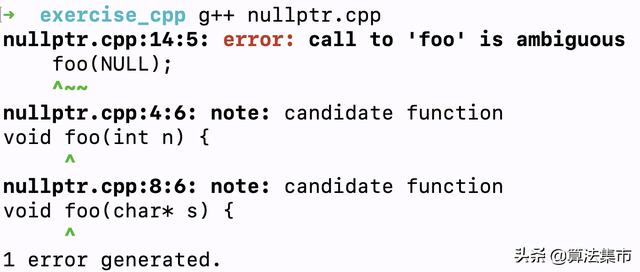
比如有下面两个函数时：

* void foo(int n)
* void foo(char\* s)

函数重载：C++允许在同一作用域中声明多个类似的同名函数，这些同名函数的形参列表（参数个数，类型，顺序）必须不同。

#include <iostream>  
using namespace std;  
  
void foo(int n) {  
 cout << "foo(int n)" << endl;  
}  
  
void foo(char\* s) {  
 cout << "foo(char\* s)" << endl;  
}  
  
int main()  
{  
 foo(NULL);  
 return 0;  
}

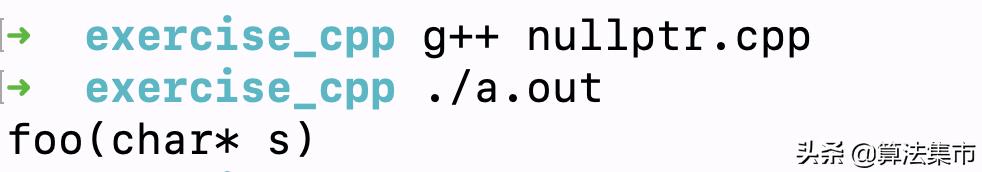
编译上述代码，结果如下图所示，编译器提示有两个函数都可能匹配，产生二义性。



而用nullptr，编译器则会选择 foo(char\* s)的函数，因为nullptr不是整数类型。

#include <iostream>  
using namespace std;  
  
void foo(int n) {  
 cout << "foo(int n)" << endl;  
}  
  
void foo(char\* s) {  
 cout << "foo(char\* s)" << endl;  
}  
  
int main()  
{  
 foo(nullptr);  
 return 0;  
}

运行结果如下图所示：



**因此，当需要使用空指针时，优先使用nullptr，而非NULL。**