

60 Why I don't 'using namespace std'

为什么我个人不使用using namespace std，这只是一种观点。

1. 什么是using namespace

就像上文中的代码用了很多标准库的内容，如果在代码前面加上

```
using namespace std;
```

C++

就可以直接写vector，find_if了，看上去代码更干净一点。

还可以把它限制在作用域中，比如写到main函数的第一行，这样main函数中调用标准库就不用写“std::”了。

所以using namespace可以非常有用，如果你在处理很长的命名空间，或是你有自己的命名空间，自己的项目文件中的符号都在这个命名空间中，你可以使用这个。

但是我个人不喜欢using namespace std

2. 为什么不喜欢

第一眼看上去代码是干净了，但是如果看原始代码，可以发现你很容易就能指出代码中使用的是C++标准模板库（带有std前缀的）。如果用了using namespace std，就相对而言有点难分辨了。如果你也用标准库喜欢用的snake case（蛇形命名法，如find_if），就很难区分到底是不是std中的。

展示一个生活中的真实例子吧。Cherno当时在EA的寒霜引擎部门工作，它们使用的是EASTL（C++ STL的替换），用的是eastl的命名空间，那如果你只写vector，我怎么知道这是在用eastl::vector还是std::vector呢？

```
namespace apple{

    void print(const std::string& text)
    {
        std::cout << text << std::endl;
    }
}

using namespace apple;
int main()
{
    // apple::print("Hello");
    print("Hello");
}
```

C++

又有一个新的命名空间orange，其中也有一个print函数，不过是打印反转的内容。

```


namespace orange{
    void print(const char* text)
    {
        std::string temp = text;
        std::reverse(temp.begin(), temp.end());
        std::cout << temp << std::endl;
    }
}

using namespace apple;
using namespace orange;

int main()
{
    print("Hello");
}

```

那这里哪一个会被调用呢？



```

int main()
{
    print(text: "Hello");
}

#ifdef 0
std::vector<int> valu

```

这并不是orange在apple后导致的，而是因为其它原因。“Hello”其实是一个const char[]数组，而不是一个string，如果只有apple命名空间，会在apple::print()中做一个隐式转换，将const char数组转换为string对象。但是引入orange命名空间后，orange::print()匹配度更高，因为它的参数本来就是一个const char*，不需要隐式转换。

如果我们不用 `using namespace`，而是简单地引入另一个库 `apple::print()` 就不会有这样的运行时错误。

另一个要百分百避免的就是在头文件中使用using namespace，永远不要这样做，把这些命名空间用在了你原本没有打算用的地方，谁知道它会include什么呢？任何大型项目中追踪起来都是很困难的，所以**绝对不要在头文件中使用using namespace!**