const

一、前言

`const` 限定符，一般来说，const 的意思就是告诉你说，对不起，你可以调用我进行计算，但是并不能够改变。这是一个很好的性质，尤其是在引用时，我们通过`const` 一方面避免了使用 引用带来的数据被篡改的危险，另一方面，又使得我们可以获得引用的效率，真的非常舒服。但是，`const` 在类中，尤其是在函数后面进行对于函数的限定的时候，情况就变得复杂了起来，这也是我们今天所要讨论问的内容。

二、函数之后的const

其实原理也很简单，就是const之后的 ，我们不会改变通过 `public` 函数改变类的 私有数据。这样，我们就不用担心类的数据被改变。这样的关键字的使用场景，主要在于 `conts complex a` .这样的我们定义了一个不可以更改的数据。但是假如我们使用一个普通的public函数，不加 `const` 的话，就有可能出现问题：因为这样的public函数是有可能更改数据的，和原始的 const 限定产生了冲突，所以就会出现不能调用函数的情况。

因此，对于那些没有更改原始数据的函数，我们一定要要在函数后面加上 const ，这样，在之后的调用中才不会出问题。下面给出示例。

三、示例

错误代码：

```
#include<iostream>
using namespace std;
class complex {
private:
    double re;
    double im;
public:
    complex(const double&r = 0, const double&i = 0) :re(r), im(i) {};
    double real() { return this->im; };
    double imag() { return this->re; };
};
int main() {
    const complex  a(0,1);
    cout << a.imag() << endl;
    system("pause");
    return 0;
}
```

会有报错：

正确代码：

```
#include<iostream>
using namespace std;
class complex {
private:
    double re;
    double im;
public:
    complex(const double&r = 0, const double&i = 0) :re(r), im(i) {};
    double real()const { return this->im; };
    double imag()const { return this->re; };
};
int main() {
    const complex  a(0,1);
    cout << a.imag() << endl;
    system("pause");
    return 0;
}
```