C++函数前和函数后加const修饰符区别

c++中关于const的用法有很多，const既可以修饰变量，也可以函数，不同的环境下，是有不同的含义。今天来讲讲const加在函数前和函数后面的区别。比如：

```
#include<iostream>

using namespace std;

// Ahthor: 过往记忆
// E-mail: wyphao.2007@163.com
// Blog:
// 转载请注明出处

class TestClass {
public:
 size_t length() const;
 const char* getPContent();
 void setLengthValid(bool isLengthValid);
private:
 char *pContent;
  size_t contentLength;    //A
  bool lengthIsValid;    //B
  size_t precontentLength;
};

size_t TestClass::length() const{ //函数名后加const
 if(!lengthIsValid){
  contentLength= strlen(pContent);  //C
  lengthIsValid = true;      //D
 }

 return contentLength;
}

const char* TestClass::getPContent(){//函数名前加const
 return pContent;
}

void TestClass::setLengthValid(bool isLengthValid){
 lengthIsValid = isLengthValid;
}
```

```
int main(void){
 TestClass *tc =new TestClass;
 tc->setLengthValid(false);
 tc->length();
 char * content = tc->getPContent();    //E
 return 0;
}
```

其中类TestClass中的length函数和getPContent函数分别在函数名后和前加了const修饰符，如果试图编译上面的代码，将会得到下面的错误:

```
--------------------配置: mingw5 - CUI Debug, 编译器类型: MinGW--------------------
检查文件依赖性...
正在编译 C:₩Users₩wyp₩Desktop₩未命名1.cpp...
[Error] C:₩Users₩wyp₩Desktop₩未命名1.cpp:24: error: assignment of data-
member `TestClass::contentLength' in read-only structure
[Error] C:₩Users₩wyp₩Desktop₩未命名1.cpp:25: error: assignment of data-
member `TestClass::lengthIsValid' in read-only structure
[Error] C:₩Users₩wyp₩Desktop₩未命名1.cpp:43: error: invalid conversion from `const char
*' to `char*'
[Warning] C:₩Users₩wyp₩Desktop₩未命名1.cpp:45:2: warning: no newline at end of file

构建中止 未命名1: 3 个错误, 1 个警告
```

里面有三个错误，也就是代码C、D、E处的三个地方。为什么C和D处的代码会出错，原因如下：length函数名的后面加了const修饰符,这样说明函数的成员对象是不允许修改的。我们都知道，在类的成员函数里面，默认是在成员函数的第一个位置是this指针，如果在成员函数(只能是成员函数，要是类的静态函数或者是非成员函数就不可以在函数名后面加上const)后面const，则说明this指针的值是不可以修改的，只能读取。而上面的length函数可能会修改里面的contentLength和lengthIsValid的值，这样编译器肯定是不允许的，所以这样是会出现错误的。
解决方法是：在类的A、B处的成员前面加上mutable修饰符：

```
mutable size_t contentLength; //A
mutable bool lengthIsValid;  //B
```

从字面的意思知道，mutalbe是“可变的，易变的”，跟constant（既C++中的const）是反义词。在C++中，mutable也是为了突破const的限制而设置的。被mutable修饰的变量，将永远处于可变

的状态，即使在一个const函数中。这样在C、D处将不会出错。
那么，为什么E处出现了错误。这是因为在函数名getPContent前加了const修饰符，意味着该函数返回的值只能是读取，而不能被修改。而E处的content却为char
*是可以被修改的，这与const正好相反了，所以出现了错误。解决方法是：在char
*前面加上const修饰符，即：

```
const char * content = tc->getPContent(); //E
```

再去编译运行，这样就不会出现错误了。