**Effective C++ Tips**

**1 基本规则**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1-1: 使用 reference to pointer，而非pointer to pointer**

说明: 如果想在函数中更改传入指针的值，则需要用到指针的指针；但是多重指针 容易出错，所以应该使用等价且清晰性更佳的指针的引用。

**1-2: 尽量以const和inline取代#define**

**1-3: 在撰写函数宏时，应该将所有的参数都加上小括号**

说明: #define是预处理指令，由C/C++预处理器在编译器之前进行处理，比如进 行宏替换。因此在编译期间将看不到符号宏名。

用#define声明函数宏时要注意其副作用。

将#define声明的宏常量声明为const常量。

将#define声明的函数宏声明为inline函数。

**1-4: 尽量以<iostream>取代<stdio.h>**

**1-5: 尽量以<iostream>取代<iostream.h>**

说明: 如果#include <iostream>，取得的是隐藏于namespace std内的iostream 程序库的元素；但如果#include <iostream.h>，则是在global scope中取 得那些元素，在global scope中取得那些元素可能会造成名称冲突。

**2 内存管理**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**2-1: 尽量以new和delete取代malloc和free**

**2-2: new/delete不能与malloc/free混用**

**2-3: 使用相同形式的new和delete**

说明: 由new分配的内存，由delete释放

由array new分配的内存，由delete[]释放

**2-4: delete空指针是安全的，什么也不会做**

**2-5: 记得在析构函数中以delete对付pointer members**  说明: 每加入一个pointer member时，几乎总是需要配合的做以下每一件事情:

·在每一个构造函数的成员初始化列表中将该指针初始化为NULL或有效值

·在赋值运算符中将指针原有的内存释放，并重新配置一块

·在析构函数中delete这个指针

**2-6: 为内存不足的状况预做准备**

说明: 使用std::set\_new\_handle为::operator new内存配置失败指定错误处理 函数（new-handle）。

一个设计良好的new-handle函数必须完成以下事情之一:

1、让更多的内存可用，使下一次的operator new成功

2、安装一个不同的new-handle

3、不安装任何new-handle，operator new则会在内存配置失败时抛出一个 std::bad\_alloc异常

4、抛出一个异常，类型为std::bad\_alloc或其派生类

5、直接调用abort或exit终止程序执行