Thinking in C++ 学习笔记

# 第二卷 使用编程技术

## 第二章 防御性编程

### 断言

断言只能用于不变量条件检查，而不能将其用于运行时错误处理。

### 单元测试

先编写单元测试程序是一种能够确保将要编写的代码能够正确工作的最好方法。能够帮助程序员将精力集中于所要完成的任务上。先编写单元测试案例然后编写代码比直接编写代码更快，同样能够防止边界条件破坏程序，使程序的代码更加健壮。

通过编写单元测试程序，开发者能够对下面两点关键内容获得足够的信心：

1. 我理解需求。
2. 我的代码符合需求。

如果得不到一个可供参照的系统，然后在它的基础上提出改进意见，人们无法清晰的说明软件需求。最好应该确定一个小的系统，设计、编写、测试这个小系统。在提出改进意见后，再重新完成它。以迭代的方式开发程序。

### 调试技术

最好的调试技术是使用断言：使用断言可以在程序代码真正出现问题之前，帮助程序设计人员找到其中的逻辑错误。

#### 用于代码跟踪的宏

**#define TRACE(ARG) cout << #ARG << endl; ARG**

这个宏在程序执行过程中将执行到的每一条语句行代码打印到cout。

**#define D(a) cout << #a “=[“ << a << “]” << endl;**

这个宏在程序执行过程中显示一个表达式，接着显示表达式的值（假设这个表达式的结果值类型重载了运算符<<）。