

## 断言(assert)的用法

分类 [编程技术](#)

我一直以为 assert 仅仅是个报错函数，事实上，它居然是个宏，并且作用并非"报错"。

在经过对其进行一定了解之后，对其作用及用法有了一定的了解，assert() 的用法像是一种"契约式编程"，在我的理解中，其表达的意思就是，程序在我的假设条件下，能够正常良好的运作，其实就相当于一个 if 语句：

```
if(假设成立)
{
    程序正常运行；
}
else
{
    报错&&终止程序！（避免由程序运行引起更大的错误）
}
```

但是这样写的话，就会有无数个 if 语句，甚至会出现，一个 if 语句的括号从文件头到文件尾，并且大多数情况下，我们要进行验证的假设，只是属于偶然性事件，又或者我们仅仅想测试一下，一些最坏情况是否发生，所以这里有了 assert()。

assert 宏的原型定义在 assert.h 中，其作用是如果它的条件返回错误，则终止程序执行。

```
#include "assert.h"
void assert( int expression );
```

assert 的作用是现计算表达式 expression，如果其值为假（即为0），那么它先向 stderr 打印一条出错信息,然后通过调用 abort 来终止程序运行。

使用 assert 的缺点是，频繁的调用会极大的影响程序的性能，增加额外的开销。

在调试结束后，可以通过在包含 #include 的语句之前插入 #define NDEBUG 来禁用 assert 调用，示例代码如下：

```
#include
#define NDEBUG
#include
```

### 用法总结与注意事项

#### 1)在函数开始处检验传入参数的合法性

如:

```
int resetBufferSize(int nNewSize)
{
    //功能:改变缓冲区大小,
    //参数:nNewSize 缓冲区新长度
    //返回值:缓冲区当前长度
    //说明:保持原信息内容不变 nNewSize<=0表示清除缓冲区
    assert(nNewSize >= 0);
    assert(nNewSize <= MAX_BUFFER_SIZE);

    ...
}
```

#### 2)每个assert只检验一个条件,因为同时检验多个条件时,如果断言失败,无法直观的判断是哪个条件失败

不好:

```
assert(nOffset>=0 && nOffset+nSize<=m_nInfomationSize);
```

好:

```
assert(nOffset >= 0);
assert(nOffset+nSize <= m_nInfomationSize);
```

#### 3)不能使用改变环境的语句,因为assert只在DEBUG个生效,如果这么做,会使用程序在真正运行时遇到问题

错误: **assert(i++ < 100)**

这是因为如果出错，比如在执行之前i=100,那么这条语句就不会执行，那么i++这条命令就没有执行。

正确:

```
assert(i < 100)
i++;
```

#### 4)assert和后面的语句应空一行,以形成逻辑和视觉上的一致感

#### 5)有的地方,assert不能代替条件过滤

程序一般分为Debug 版本和Release 版本，Debug 版本用于内部调试，Release 版本发行给用户使用。断言assert 是仅在Debug 版本起作用的宏，它用于检查"不应该"发生的情况。以下是一个内存复制程序，在运行过程中，如果assert 的参数为假，那么程序就会中止（一般地还会出现提示对话，说明在什么地方引发了assert）。

以下是使用断言的几个原则：

- （1）使用断言捕捉不应该发生的非法情况。不要混淆非法情况与错误情况之间的区别，后者是必然存在的并且是一定要作出处理的。
- （2）使用断言对函数的参数进行确认。
- （3）在编写函数时，要进行反复的考查，并且自问："我打算做哪些假定？"一旦确定了的假定，就要使用断言对假定进行检查。
- （4）一般教科书都鼓励程序员们进行防错性的程序设计，但要记住这种编程风格会隐瞒错误。当进行防错性编程时，如果"不可能发生"的事情的确发生了，则要使用断言进行报警。

ASSERT ()是一个调试程序时经常使用的宏，在程序运行时它计算括号内的表达式，如果表达式为FALSE (0), 程序将报告错误，并终止执行。如果表达式不为0，则继续执行后面的语句。这个宏通常原来判断程序中是否出现了明显非法的数据，如果出现了终止程序以免导致严重后果，同时也便于查找错误。

ASSERT 只有在 Debug 版本中才有效，如果编译为 Release 版本则被忽略。

原文地址：<https://www.cnblogs.com/thisway/p/5558914.html>

← Python 函数参数前面一个星号 (\*) 和两个星号 (\*\*) 的区别

结构体大小的计算 →

✍ 点我分享笔记

### 教程列表

ADO 教程	Ajax 教程	Android 教程
Angular2 教程	AngularJS 教程	AppML 教程
ASP 教程	ASPNET 教程	Bootstrap 教程
Bootstrap4 教程	Bootstrap5 教程	C 教程
C# 教程	C++ 教程	CSS 参考手册
CSS 教程	CSS3 教程	Django 教程
Docker 教程	DTD 教程	ECharts 教程
Eclipse 教程	Firebug 教程	Font Awesome 图
Foundation 教程	Git 教程	Go 语言教程
Google 地图 API 教	Highcharts 教程	HTML DOM 教程
HTML 参考手册	HTML 字符集	HTML 教程
HTTP 教程	ionic 教程	iOS 教程
Java 教程	JavaScript 参考手	Javascript 教程
jQuery EasyUI 教程	jQuery Mobile 教程	jQuery UI 教程
jQuery 教程	JSON 教程	JSP 教程
Julia 教程	Kotlin 教程	Linux 教程
Lua 教程	Markdown 教程	Matplotlib 教程
Maven 教程	Memcached 教程	MongoDB 教程
MySQL 教程	Node.js 教程	NumPy 教程
Pandas 教程	Perl 教程	PHP 教程
PostgreSQL 教程	Python 3 教程	Python 基础教程
R 教程	RDF 教程	React 教程
Redis 教程	RSS 教程	Ruby 教程
Rust 教程	Sass 教程	Scala 教程
SciPy 教程	Servlet 教程	SOAP 教程
SQL 教程	SQLite 教程	SVG 教程
SVN 教程	Swift 教程	TCP/IP 教程
TypeScript 教程	VBScript 教程	Vue.js 教程
Vue3 教程	W3C 教程	Web Service 教程
WSDL 教程	XLink 教程	XML DOM 教程
XML Schema 教程	XML 教程	XPath 教程
XQuery 教程	XSLFO 教程	XSLT 教程
数据结构	正则表达式	测验
浏览器	网站品质	网站建设指南
网站服务器教程	设计模式	

#### 在线实例

- [HTML 实例](#)
- [CSS 实例](#)
- [JavaScript 实例](#)
- [Ajax 实例](#)
- [jQuery 实例](#)
- [XML 实例](#)
- [Java 实例](#)

#### 字符集&工具

- [HTML 字符集设置](#)
- [HTML ASCII 字符集](#)
- [HTML ISO-8859-1](#)
- [PNG/JPEG 图片压缩](#)
- [HTML 拾色器](#)
- [JSON 格式化工具](#)

#### 最新更新

- [Julia 数据类型](#)
- [Julia 元组](#)
- [Python2 与 Pyth...](#)
- [Julia 数组](#)
- [Python3 reload\(...](#)
- [Julia 基本语法](#)
- [Julia 交互式命令](#)

#### 站点信息

- [意见反馈](#)
- [免责声明](#)
- [关于我们](#)
- [文章归档](#)

#### 关注微信

