适合PC用户和C/C++程序开发人员阅读

C函数库·C++类库使用手册

韩 滨 魏海萍 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

C函数库·C++类库使用手册

韩 滨 魏海萍 编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书旨在全面介绍C函数库和C++类库，并提供相关的编程示例。全书分为两大部分，共19章。第一篇简要比较C89与C99标准的差异，并全面介绍最新的标准C函数库，包括I/O、字符与字符串、数学、动态分配、实用工具、宽字符等函数，以及C99新增的库特征。第二篇简要讨论标准C++的面向对象特性与标准模板库，并全面介绍最新的标准C++类库，包括I/O类、STL容器类、STL算法、字符串类、数学类、异常处理类等。

本书叙述清晰，内容全面，是广大PC用户和C/C++程序开发人员的一部很有价值的工具书，也可以作为计算机软件开发人员和大专院校师生的参考书。

图书在版编目（CIP）数据

C函数库·C++类库使用手册/韩滨等编著.—北京：电子工业出版社，2004.4
ISBN 7-5053-9762-1

Ⅰ.C… Ⅱ.韩 Ⅲ.C语言－程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2004）第019532号

责任编辑：李 莹
印 刷：北京天竺颖华印刷厂
出版发行：电子工业出版社
北京市海淀区万寿路173信箱 邮编：100036
北京市海淀区翠微东里甲2号 邮编：100036
经 销：各地新华书店
开 本：787×1092 1/16 印张：22.75 字数：580千字
印 次：2004年4月第1次印刷
定 价：30.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

前　言

近 10 年来，程序设计领域发生了许多变化。Internet 和 Web 逐渐兴起，Java 语言开始流行，.NET 框架已经发布，C#已经问世，C++已进行了标准变化，新的 C 标准——C99 标准也已经制定出来。虽然 C 和 C++的标准化没有得到大肆宣称，但这确实是过去 5 年计算机领域发生的一件大事。

在历史的发展进程中，人们往往只重视新鲜事物，而忽视用来构建未来的基础。C 语言就是许多程序设计语言的基础。C++是以 C 语言为基础发展起来的，C 语言的语法构成了 Java 语言的基础。如果 C 仅仅是其他语言的起点，就会遭到历史的淘汰。幸运的是，自从 C 语言问世以来，它一直是一种很重要的程序设计语言。C 和 C++长时间的成功在程序设计语言开发的历史上留下了不可磨灭的印记。读者从本书中将会看到，C99 标准和标准 C++引进了富有创新精神的新特性，而这些新特性再次将 C 和 C++推到了程序设计的前沿。尽管 Java 逐渐流行，但 C 和 C++仍拥有其他程序设计语言所无法比拟的独特魅力。坦率地说，要想成为一名专业程序设计人员，就需要精通 C 和 C++。它们是所有程序员都不能忽视的程序设计语言。

本书是为有一定编程经验与编程水平的读者而编写的，要求读者至少能够编写简单的 C 或 C++程序。对正在学习 C 或 C++语言的读者，本书是任何 C 或 C++语言教程的最佳配套参考书。对正在编写应用程序的读者，本书是一本不可多得的工具书。

本书全部描述了标准 C 函数库和标准 C++类库，包括标准 C 和 C++中新引进的各种特性。全书分为两个部分：第一篇主要介绍标准 C 函数库，第二篇主要介绍标准 C++类库。

第一篇共有 10 章，分别介绍 C 的新增特性和 C 的各种库函数，并提供适当的编程示例。其中主要包括如下内容：

C89 与 C99 标准的比较；

I/O 函数；

字符与字符串函数；

数学函数；

时间与日期函数；

动态分配函数；

实用工具函数；

宽字符函数；

C 中的新增库函数特性；

自定义函数。

第二篇共有 9 章，分别介绍标准 C++的新增特性和标准 C++类库，并给出适当的编程示例。其中主要包括如下内容：

标准 C++的面向对象特性和标准模板库；

I/O 类；

STL 容器类；

STL 算法；

STL 分配器、迭代器和函数对象；

字符串类；

数字类；

异常处理类。

本书的第一篇由韩滨编写，第二篇由魏海萍编写。本书的编写思想立足于函数和类的介绍，因此力求简洁，减少了基础知识的介绍。由于编者水平有限，加上时间仓促，书中难免有疏漏或欠妥之处，殷切期望广大读者及同行批评指正。

目　录

第一篇　标准 C 函数库

第二篇 标准 C++类库

第一篇　标准C函数库

标准 C 定义了内容丰富的函数库，提供对输入/输出、字符与字符串处理、数学运算、时间与日期处理等常见操作的支持。

标准 C 函数库分成以下几类：

输入/输出函数

字符与字符串处理函数

数学函数

时间、日期与本地化函数

宽字符函数

最后一类函数于 1995 年被引进到标准 C 中，后来又被合并到 C++中。这类函数提供了与几个库函数具有相同功效的宽字符。

C99 标准给 C 函数库添加了一些新元素，其中的几个（如支持复数运算、通用型数学函数宏）借鉴了 C++中的已有特性。C99 新增的库元素与 C++是不兼容的。

最后需要指出的是，所有编译程序所提供的函数都比标准 C/C++所定义的函数多，通常情况下，这些附加函数用来支持操作系统接口和环境特有的其他操作。具体细节应该查阅编译程序的用户手册。

第 1 章　C89 与 C99 标准的比较

随着新的计算机技术、方法论、硬件以及软件开发需求的不断发展和提高，诸如 C 这样的程序设计语言也不断进步。C 语言的进步途径有两条：基于 C 发展 C++，以及不断发展与完善 C 自身。

在过去几年中，C 的发展重点主要放在 C++上，但是 C 自身的不断改进也从来没有间断过。随着计算环境的全球化，原来的 C89 标准于 1995 年做过一次修订，进而引进了各种宽字符与多字节功能。随后，C 语言的更新与完善工作逐渐展开，进而形成了当今的新标准：C99。在新的 C99 标准中，C 的关键元素几乎没有发生变化，即使有，也只是对 C 所做的少量改进，并增加了部分新的库函数。

1.1　C99 综述

与 C89 标准相比，C99 标准中主要有三个方面的变化：

新增了 C89 标准中所没有的部分特性；

废弃了 C89 标准中的部分特性；

改进或增强的特性。

一般说来，C89 与 C99 标准之间的大多数特性没有发生太大变化，所以 C 也只存在细微差别。下面将着重介绍 C99 中对程序设计方式有较大影响的特性。

1. 废弃的特性

C89 标准支持的“隐含式 int”规则在 C99 标准中不再受支持。在 C89 标准中，如果没有显式地声明变量的类型，该变量的类型将被当做 int 类型来对待。如果在使用函数之前没有声明它，它将被当做隐含式函数来对待。需要注意的是，在将代码移植到 C99 标准时，涉及这两个特性的代码需要改写。

2. 新增与改变的特性

在 C99 标准中，最值得一提的新增特性是如下这些关键字：

inline；

restrict；

_Bool；

_Complex；

_Imaginary。

C99 标准中的其他新增特性还包括：

支持可变长数组；

支持复数；

支持 long long int 数据类型；

支持“//”格式的单行注释；

支持分散声明与数据的处理；

新增预处理程序；

新增 for 语句中的变量声明；

支持复合赋值；

新增柔性数组结构成员；

新增指定的初始化符；

修改 printf()和 scanf()函数系列；

新增__func__预定义标识符；

新增库和头部文件。

在 C99 标准中，增加的大部分特性均是新的创新，其中的大多数基于各种 C 工具所提供的语言扩充，但有些特性借鉴了 C++等程序设计语言，比如 inline 关键字和“//”格式的单行注释。请注意，C99 标准只是借鉴，而不是加进了 C++类、继承性或成员方法等面向对象特性，目的是为了保持 C 的本色不变。

改变的重要特性包括：

放宽的转换限制；

扩充的整数类型；

增强的整数类型提升规则；

对 return 语句的约束。

由于对 return 语句的改变将影响现有应用程序，因此需要对涉及这项改变的代码进行改写，以便代码能兼容于 C99 标准。

下面将重点介绍 C89 与 C89 标准之间的主要差别。

1.1.1　restrict 指针

在 C99 标准中，最重要的创新之一是仅适用于指针的 restrict 类型修饰符。restrict 指针是初始访问由该指针所指对象的惟一途径，因此只有借助于 restrict 指针表达式才能访问对象。restrict 指针主要用做函数变元，或者指向由 malloc()函数所分配的内存变量。restrict 数据类型不改变程序的语义。

编译程序通过假设 restrict 指针是访问对象的惟一途径，更好地优化了某些类型的例程。例如，如果某个函数定义了两个 restrict 指针变元，编译程序就假定它们指向两个不同的对象，memcpy()函数就是 restrict 指针的一个典型应用示例。在 C89 标准中，memcpy()函数的原型如下：

```
void *memcpy(void *s1, const void *s2, size_t size);
```

如果 *s1* 和 *s2* 所指向的对象重叠，其操作行为就是未定义的。因此，memcpy()函数只能用于不重叠的对象。

在 C99 标准中，可以在 memcpy()原型中显式地使用 restrict 修饰符。例如，在 C99

标准中，memcpy()函数的原型如下：

void *memcpy(void *restrict *s1*, const void *restrict *s2*, size_t *size*);

通过使用 restrict 修饰 *s1* 和 *s2* 变元，可确保它们在该原型中指向不同的对象。由于 restrict 修饰符具有潜在优点，C99 标准已经把它加入到许多 C89 库函数的原型中。

1.1.2 inline 关键字

C99 标准中新引进了关键字 inline【内联】。该关键字适用于函数，将 inline 关键字放在函数声明的前面，意味着告诉编译程序需要优化函数调用，即让编译程序将函数的代码联机扩展，而不是调用它。编译程序有可能不兑现 inline。C99 标准规定，inline 的作用仅仅是提醒编译程序，尽可能使函数调用的运行速度加快。inline 关键字在 C++中也得到了支持，而且在 C99 中的语法与 C++中的完全一致。如果希望联机扩展函数，可以在函数名的前面放上 inline 关键字：

```
inline int sum(int k, int m, int n)
{
  // calculate the sum of k, m and n variables
}
```

内联函数十分重要，因为它们除了保持结构化以及函数式的定义方式之外，还能使程序员编写出高效率的代码。函数的每次调用与返回都会消耗相当大的系统开销，尤其是当函数调用发生在重复次数很多的循环语句中时。一般情况下，当发生一次函数调用时，变元需要进栈，各种寄存器内存需要保存。当函数返回时，寄存器的内容需要恢复。如果该函数在代码内进行联机扩展，当代码执行时，这些保存和恢复操作就不会再发生，而且函数调用的执行速度也会大大加快。

需要注意的是，尽管函数的联机扩展能够产生较快的执行速度，但这种扩展方式会产生较长的代码，所以只应该内联对应用程序性能有显著影响的函数以及长度较短的函数。此外，编译程序可能会忽略这种联机扩展请求，或者采取其他方式来优化函数调用。

1.1.3 新增的数据类型

C99 标准新引进了下列内部数据类型：

_Bool；

_Complex 和_Imaginary；

long long int。

1. _Bool

C99 标准中新引进了_Bool 数据类型。_Bool 是一个整数类型，其值可以是 1 或 0（分别表示“真”和“假”），并且同 C++中所定义的 bool 关键字不同。因此，C99 和 C++在这方面是不兼容的。C++中定义了内部布尔常数 true 和 false，而 C99 中则增加了用来定义 bool、true 及 false 宏的头部文件<stdbool.h>，以便程序员能够编写同时兼容于 C 与 C++的应用程序。

C99 标准中使用_Bool 而不是 bool 作为关键字，目的是为了避免破坏现有代码，因为许多现有的 C 应用程序已经自定义了它们自己的 bool 版本。因此，在编写新的应用程序时，应该使用<stdbool.h>头部文件中的 bool 宏。

2. _Complex 与_Imaginary

通过包含相应的头部文件和几个新的库函数，C99 标准中新引进了复数运算支持，目的是为了给进行数值计算的应用程序提供更充分的支持。但是，实现虚数不需要任何工具，包括独立于操作系统的外部工具。C99 标准中定义的复数类型如下所示：

```
float _Complex;
float _Imaginary;
double _Complex;
double _Imaginary;
long double _Complex;
long double _Imaginary。
```

同样，使用_Complex 和_Imaginary 而不是 complex 和 imaginary 作为关键字，目的是为了避免破坏现有代码，因为许多现有的 C 应用程序已经自定义了它们自己的 complex 和 imaginary 数据类型。<complex.h>头部文件中定义了 complex 和 imaginary 宏，并将它们扩展为_Complex 和_Imaginary，因此在编写新的应用程序时，应该使用<stdbool.h>头部文件中的 complex 和 imaginary 宏。

3. long long int 数据类型

C99 标准中新引进了 long long int 和 unsigned long long int 数据类型。long long int 类型的最小取值范围从-（2^{63}-1）至 2^{63}-1。unsigned long long int 类型的最小取值范围从 0 至 2^{64}-1。long long int 数据类型能够支持的整数长度为 64 位。

1.1.4　对数组的增强

C99 标准中新增了两个重要的数组特性：可变长数组以及数组声明中的类型限定。

1. 可变长数组

C89 标准规定，必须将数组的维数声明成整型常数，并且数组的大小在编译时必须是确定的。在 C99 标准中，情况有所变化，程序员声明数组时，数组的维数可以由任一有效的整型表达式来确定，包括只在运行时才能确定其值的表达式，这类数组就叫做可变长数组。但是，只有局部数组才可以是变长的。

需要注意的是，可变长数组的维数在数组生存期内是不变的，也就是说，可变长数组不是动态的。可以变化的只是数组的大小。

程序员可以使用*来定义不确定长的可变长数组。

C99 标准中引进可变长数组的重要原因之一是支持数值处理，而且这也是一个十分

实用的特性，但C89和C++不支持这个特性。

2. 数组声明中的类型修饰符

在C99标准中，如果需要使用数组作为函数变元，则可以在数组声明的方括号内使用static关键字，这相当于告诉编译程序，变元所指向的数组将至少包含指定的元素个数。例如：

```
int example(char arrary[static 20]);
```

其中，arrary变元指向该数组的起点，并且该数组至少包含80个元素。

程序员也可以在数组声明的方括号内使用restrict、volatile和const关键字，但只用于函数变元。如果使用restrict关键字，指针是初始访问该对象的惟一途径。如果使用const关键字，指针始终指向同一个数组。使用volatile关键字没有任何意义。

1.1.5 单行注释

C99标准新引进了单行注释。这种注释以“//”开头，直至当前行结束。例如：

```
//This is a single comment line
int a;  //this is an int variable
```

1.1.6 分散代码与声明

C89标准规定，同一个模块内的所有声明必须放在第一条可执行代码之前，但C99标准中取消了这项规定，程序员可以根据需要在模块内的任何地方随时声明变量。由于C++广泛使用了分散声明与代码的特性，故C99标准中引进这一特性极大地方便了程序员编写适用于这两种环境的代码。

1.1.7 预处理程序的修改

C99标准对预处理程序进行了少量小改动。

1. 变元列表

预处理程序的最大改动是宏可以带变元，在宏定义中用省略号（...）表示。内部预处理标识符__VA_ARGS__决定变元将在何处得到替换。例如，假设有下列定义：

```
#define MySum(...) sum(__VA_ARGS__)
```

那么语句

```
MySum(k, m, n);
```

将被转换成如下形式：

```
sum(k, m, n);
```