C++ 大全

（2.1 版）

张玉亭 韩 兰 木 林 编译

北京希望电脑公司

C++ 大 全

（2.1版）

张玉亭　　韩兰　　木林　编译

北京希望电脑公司

一九九一、十二

★北京市新闻出版局
准印证号：3329-90321

★订购单位：北京8721信箱资料部

★邮　　码：100080

★电　　话：2562329

★传　　真：01—2561057

★乘　　车：320、302、332、路车
到海淀黄庄下车

★办公地点：希望公司大楼一楼往里走
101房间

前　言

近年来，面向对象编程（OOP）被认为是编程方面的一场实质性革命，面向对象编程方法已得到科研人员的深入研究，并在广大程序员中广泛使用。

C++作为一种语言运用了OOP编程的关键成份，使程序员可以使用某些强功能的面向对象编程的设计方法。C++的目标是编写一种程序员很快便可以使用的语言，并用这一语言进一步扩大其编程知识，增加其编程能力、提高其编程水平，而不是从头学习一种新的语言。尽管C++已达到了这一目的，且已对程序编写的方法产生了深刻的影响，但是，如果用与C语言相同的编程方法来编写C++程序的话，那么，这本身就意味着失败。这并不是说如此编写的程序不能运行，而是说如此编写的程序未能充分地利用C++。

我们向读者提供这本有关C++的技术参考大全，通过对有关概念的深入讲解，以及大量列举编程实例，使读者能更全面、更详细地了解C++的特性，从而为读者在以后的编程实践中充分地利用这些特性打下坚实的基础，最终编写出优秀地的C++程序。

在本书的出版过程中，得到了北京希望电脑公司资料部秦人华经理、杨淑欣老师的热情帮助和大力支持，在此表示衷心的感谢。

编　译　者

一九九一年十二月

简　介

自C语言问世以来，C++是编程艺术和科学中的最重要进展。毫无疑问，C++将改变人们编写程序的方式。

在编程方面，C++向前迈出了重要一步。基于C语言，C++所添加的扩展支持面向对象编程。这些扩展戏剧性地增强了C语言的性能。这也就是说，C语言发展变化的下一步便是C++。

面向对象编程的技能使程序员能管理好不断变大且更为复杂的程序。最终，C++会获得圆满成功。

不过，C++不仅仅适用于面向对象编程，还因其先进特性，它使得编写非面向对象程序也更为容易。理解这一点是非常重要的。

在编写程序时，C语言是专业程序员所选择的语言。由于C++是C语言的增强版本，所以可以预料，在今后几年C++的普及会突飞猛进地增长。事实上，许多工业分析人员预测，九十年代将是C++的年代。坦率地说，C++的功能和通用性，加之已普及的C语言定会使其大获成功。

本书涉及C++的C式概念和C++专用特性。然而，更注重于C++的特性。

目　录

第一部分　C++基础；C式特性

第一章　C概述

1.1　C 起源 …… (1)
1.2　C 是中级语言 …… (1)
1.3　C 是结构化语言 …… (2)
1.4　C 是程序员的语言 …… (3)
1.5 C程序的格式 …… (4)
1.6 库和链接 …… (5)
1.7 分别编译 …… (6)

第二章　表达式

2.1　五个基本数据类型 …… (6)
2.2　修改基本类型 …… (6)
2.3　标识符名 …… (7)
2.4　变量 …… (8)
2.5　存取类型修饰符 …… (12)
2.6　存储类修饰符 …… (13)
2.7　变量初使化 …… (17)
2.8　常量 …… (17)
2.9　算符 …… (19)
2.10　表达式 …… (30)

第三章　C语句

3.1　C中的真与假 …… (33)
3.2　选择语句 …… (33)
3.3　迭代语句 …… (41)
3.4　转移语句 …… (47)
3.5　表达式语句 …… (51)
3.6　块语句 …… (51)

第四章　数组和串

4.1　一维数组 …… (52)
4.2　生成指向数组的指针 …… (53)
4.3　将一维数组传送给函数 …… (53)
4.4　串 …… (54)
4.5　二维数组 …… (55)

4.6 多维数组 …… (59)
4.7 指针下标 …… (59)
4.8 数组初始化 …… (61)
4.9 实例：Tic-Tac-Toe游戏 …… (62)
第五章 指针
5.1 什么是指针 …… (65)
5.2 指针变量 …… (65)
5.3 指针算符 …… (66)
5.4 指针表达式 …… (67)
5.5 指针和数组密切相关 …… (69)
5.6 多级间址 …… (71)
5.7 初始化指针 …… (72)
5.8 指向函数的指针 …… (73)
5.9 C的动态分配函数 …… (74)
5.10 指针带来的问题 …… (76)
第六章 函数
6.1 函数的一般格式 …… (78)
6.2 函数的作用域规则 …… (78)
6.3 函数变元 …… (79)
6.4 argc和argv——main ( ) 的变元 …… (82)
6.5 return语句 …… (84)
6.6 返回非整型值的函数 …… (86)
6.7 函数原型 …… (87)
6.8 返回指针 …… (88)
6.9 void类型函数 …… (89)
6.10 main ( ) 返回什么 …… (90)
6.11 递归 …… (90)
6.12 说明变量长度参数列表 …… (91)
6.13 古典与现代函数参数说明 …… (91)
6.14 实现事宜 …… (92)
第七章 结构，联合，枚举和用户定义类型
7.1 结构 …… (93)
7.2 结构数组 …… (96)
7.3 将结构传送给函数 …… (96)
7.4 结构指针 …… (97)
7.5 结构内的数组和结构 …… (99)
7.6 位字段 …… (100)
7.7 联合 …… (101)
7.8 枚举 …… (103)

7.9 使用sizeof，确保可移植性 ……（105）
7.10 typedef ……（105）
第八章 控制台I/O
8.1 应用注释 ……（106）
8.2 读写字符……（107）
8.3 读写串……（108）
8.4 格式化控制台I/O ……（110）
8.5 printf（） ……（110）
8.6 scanf（） ……（115）
第九章 ANSI C标准文件I/O
9.1 历史注释……（119）
9.2 流和文件……（119）
9.3 文件系统基础……（120）
9.4 fread（）和fwrite（） ……（128）
9.5 fseek（）和随机存取I/O ……（130）
9.6 fprintf（）和fscanf（） ……（131）
9.7 标准流……（131）
第十章 C预处理器和注释
10.1 C预处理器 ……（133）
10.2 #define ……（133）
10.3 #error……（135）
10.4 #include……（135）
10.5 #条件编译命令 ……（136）
10.6 #undef ……（139）
10.7 #line ……（139）
10.8 #pragma……（139）
10.9 #和##预处理器算符 ……（139）
10.10 预定义宏名……（140）
10.11 注释……（141）

第二部分 C++专用特性

第十一章 C++概述
11.1 C++起源 ……（142）
11.2 什么是面向对象编程 ……（143）
11.3 用C++风格编程 ……（144）
11.4 介绍C++类 ……（147）
11.5 函数重载 ……（149）
11.6 算符重载 ……（151）
11.7 继承的功能 ……（151）

11.8 构造函数和析构函数 ……（155）
11.9 C++关键字 ……（157）
11.10 C++程序的一般格式 ……（157）
第十二章 类和对象
12.1 类 ……（158）
12.2 结构和类 ……（160）
12.3 联合和类 ……（161）
12.4 友元函数 ……（163）
12.5 内部函数 ……（166）
12.6 在类中定义内部函数 ……（167）
12.7 参数化的构造函数 ……（168）
12.8 静态类成员 ……（196）
12.9 何时执行构造函数和析构函数 ……（173）
12.10 嵌套类 ……（174）
12.11 作用域分辨算符 ……（174）
12.12 局部类 ……（175）
12.13 将对象传送给函数 ……（175）
12.14 返回对象 ……（176）
12.15 对象赋值 ……（177）
第十三章 数组，指针和引用
13.1 对象数组 ……（177）
13.2 指向对象的指针 ……（179）
13.3 this指针 ……（180）
13.4 指向派生类型的指针 ……（181）
13.5 指向类成员的指针 ……（182）
13.6 引用 ……（184）
13.7 形式 ……（188）
13.8 C++的动态分配算符 ……（189）
第十四章 函数和算符重载
14.1 函数重载 ……（193）
14.2 overload ……（196）
14.3 重载构造函数 ……（196）
14.4 寻找被重载函数的地址 ……（198）
14.5 算符重载 ……（198）
14.6 使用友元的算符重载 ……（202）
14.7 重载new和delete ……（206）
14.8 重载某些特殊算符 ……（209）
第十五章 继承
15.1 基类存取控制 ……（214）

15.2 继承多重基类 …… (218)
15.3 构造函数，析构函数和继承 …… (219)
15.4 授权存取 …… (224)
15.5 虚拟基类 …… (225)
第十六章 虚函数和多态性
16.1 虚函数 …… (228)
16.2 纯虚函数 …… (233)
16.3 使用虚函数 …… (234)
16.4 前联编和后联编 …… (236)
第十七章 C++I/O系统基础
17.1 C++流 …… (237)
17.2 基本流类 …… (237)
17.3 格式化I/O …… (238)
17.4 重载<<和>> …… (246)
17.5 建立自己的操作函数 …… (251)
17.6 有关旧流类库的简要说明 …… (256)
第十八章 C++I/O文件
18.1 fstream.h和文件类 …… (256)
18.2 开启和关闭文件 …… (256)
18.3 读写文本文件 …… (258)
18.4 二进制I/O …… (259)
18.5 更多的get( )函数 …… (262)
18.6 getline( ) …… (262)
18.7 检测EOF …… (263)
18.8 ignore( )函数 …… (265)
18.9 peek( )和putback( ) …… (265)
18.10 flush( ) …… (265)
18.11 随机存取 …… (265)
18.12 I/O状态 …… (268)
18.13 定制的I/O和文件 …… (269)
第十九章 基于数组的I/O
19.1 基于数组的类 …… (271)
19.2 建立基于数组的输出流 …… (271)
19.3 把数组用做输入 …… (273)
19.4 使用二进制I/O …… (274)
19.5 基于数组的输入/输出流 …… (274)
19.6 数组内的随机存取 …… (275)
19.7 使用动态数组 …… (275)
19.8 操作程序和基于数组的I/O …… (276)

19.9 定制析取程序和插入程序 …… (277)
19.10 有关基于数组格式化的使用 …… (278)
第二十章 事宜和高级主题
20.1 缺省函数变元 …… (279)
20.2 建立转换函数 …… (281)
20.3 拷贝构造函数 …… (284)
20.4 动态初始化 …… (285)
20.5 const和volatile成员函数 …… (286)
20.6 使用asm关键字 …… (286)
20.7 链接说明 …… (287)
20.8 overload …… (287)
20.9 C与C++之间的差异 …… (287)
20.10 C++的未来发展趋势 …… (288)

第三部分 某些C++应用程序

第二十一章 串类
21.1 定义串类型 …… (289)
21.2 串类 …… (290)
21.3 构造函数和析构函数 …… (291)
21.4 有关字符串的I/O …… (292)
21.5 赋值函数 …… (293)
21.6 连接 …… (294)
21.7 子串减法 …… (296)
21.8 关系算符 …… (297)
21.9 杂项串函数 …… (298)
21.10 整个串类 …… (298)
21.11 使用串 …… (304)
第二十二章 上托窗口类
22.1 上托窗口 …… (306)
22.2 建立某些视频支持函数 …… (307)
22.3 窗口类 …… (310)
22.4 显示并删除窗口 …… (312)
22.5 窗口I/O …… (314)
22.6 整个窗口系统 …… (317)
22.7 修改 …… (325)
第二十三章 链表类
23.1 双链表类 …… (326)
23.2 建立类属双链表基类 …… (334)

23.3 其它实现 …… (338)

附录A 某些常见表

A.1 复类 …… (340)

A.2 BCD类 …… (341)

第一部分　C++基础：C式特性

本书第一部分共十章讨论C++的C式特性。正如你所知，C++是ANSI标准C的一种增强版本。因此，任意C++编译程序按定义是C编译程序。因为C++是基于C建立的，所以，除非你了解如何用C编程，否则，不能用C++编程。

进一步说，构成C基础的许多基本概念也构成了C++的基础，它们在本书这一部分加以讨论。由于C++是C的超集，所以在本部分描述的任何内容均适用于C++。C++的专用特性将在本书的第二部分详细讨论。

第一章　C 概 述

本章的目的是概述C编程语言及其起源，使用和基本结构。由于C++是基于C建立的，所以这一章对C++的根进行了重要的历史性透视。

1.1　C起源

C语言的最初设计和实现是由Dennis Ritchie在DEC PDP-11上的UNIX系统环境下完成的。C语言的前身是BCPL语言。BCPL语言是由Martin Richards设计的。它深刻地影响了Ken Thompsor设计的B语言风格。B语言于1970年发展成为C语言。

多少年来，C的标准版本一直是UNIX操作系统支持的版本。它的经典描述是Brian kernighan和Dennis Ritchie于1978年出版的《C程序设计语言》这本书提供的。随着微型机的普及，大量的C实现纷纷相继建立。由于此时无标准，因此彼此之间存在差异，且不可兼容。为了改变这种状况，美国国家标准协会（ANSI）于1983年夏初建立了一个委员会，以创建ANSI标准。ANSI C标准现已采用，且所有主要C和C++编译程序已实现这一标准。

1.2　C是中级语言

C语言一般被称为中级计算机语言。所谓计算机语言并不是指它的能力是中级的或难以使用，也不是指它不如BASIC或PASCAL之类的高级语言高级、但C语言又不象汇编语言那样经常给使用者带来麻烦。它只不过是把高级语言的一些特征同汇编语言的能力结合起来形成了所谓中级语言。下面列举出有关语言及其分类，从中可看到C在这些语言中的位置。

高级语言	Ada
	Modula-2
	Pascal
	COBOL
	BASIC
	FORTRAN
中级语言	C
	FORTH

宏汇编语言

低级语言　　汇编语言

作为一种中级语言，C允许对字节，字值和地址直接操作。而地址是计算机的基本工作单元。且C语言也很便于移植。所谓移植就是把一个软件从一类计算机转移到另一类计算机上，但仍能保持其功能。例如，如果在Apple Macintosh机上编写的程序可以在IBM PC机上运行，那么，就称这一程序是可移植的。

所有程序设计语言都接受数据类型这一概念。一个数据类型定义了一组值。通过对变量的一组操作可存储该变量。一般数据类型包括整型，字符型和实型。尽管C语言有五种基本类型，但它不象Pascal或Ada那样的语言对类型有严格的要求。C语言几乎允许所有类型相互转换。例如，在众多表达式中，C语言允许整型和字符型自由混用。而且C语言也不进行象数组越界或变量类型不匹配之类的运行错误检查。不过，在有些版本中，对除零错误和堆栈溢出错误还是提供报告的。但这类检查都由程序员负责。

C语言的特殊之处在于它允许对位、字节、字和指针直接操作。这很适合系统级编程，因为在系统级编程过程中常常要用到这些操作。C语言的另一重要方面是仅有32个关键字（其中Kerhighan和kitchie定义了27个，ANSI标准委员会增加了5个）。这些关键字构成了C语言的框架。与之对应的是IBMPC机上的BASIC语言有大约159个关键字。

1.3 C是结构化语言

尽管“块结构化语言”这一说法不严格适用于C，但C常被简单地称为结构化语言。它与ALGOL，Pascal和Modula-2这样的结构化语言有许多相同之处。从技术角度上讲，块结构化语言允许在过程或函数内部说明其它过程或函数。这样的话，全局变量和局部变量的概念扩展到使用作用域规则。该规则控制着变量或过程的可见性。因此，由于C语言不允许在函数内部建立函数，它也就不能严格地称为块结构化语言。

结构化语言的显著特性就是代码和数据的独立性。这类语言可以把完成特定任务所用的信息和指令同程序的其它部分分隔开。其方法就是使用局部变量和子程序。用局部变量可以编写出对程序的其它部分无副作用的子程序。这样，编写出需要共享的代码就很容易了。换言之，如果要研制一个独立函数，只需了解该函数要做些什么，而无需了解它怎样做。需要注意的是，过多地使用全局变量（在整个程序中起作用的变量）会导致不必要的副作用，从而使程序极易出错（使用过BASIC的程序员都会遇到这类麻烦）。

结构化语言允许在程序设计时有各种选择余地。它支持若干种循环结构，如：while，do-while，和for语句。在结构化语言中，不提倡使用goto语言，甚至禁止使用之。这与BASIC或FORTRAN语言不同，它们用goto作为一种常用的控制结构。此外，结构化语言允许使用缩进语句的书写形式，且没有严格的语句位置概念（FORTRAN则严格要求语句的位置）。

下面列出的是几个结构化语言和非结构化语言：

非结构化的	结构化的
FORTRAN	Pascal
BASIC	Ada
COBOL	C

Modula-2

结构化语言是现代的发展趋势，而非结构化的语言正在过时。实际上，旧的计算机语言的标志就是它是非结构化的。目前，绝大多数程序员认为结构化语言更易编程。

C语言的主要结构成分是函数：C语言的独立例程。在C语言中，函数是块结构，所有程序的动作都在其内部进行。C语言允许把程序的每个任务的定义和编码分别实现，从而使程序更易模块化。一旦建立函数，就可把它放在不同的环境中工作，而不必担心它会对程序的其它部分产生副作用。在大型项目中，需严格要求建立独立性很强的函数。这样，一个程序员编写的代码便不会影响到另一人编写的代码，这一点是非常重要的。

在C语言中，另一种结构化和独立化的方法是使用代码块。代码块就是指一组程序语句在逻辑上可以当作一个单位来使用。在C中，代码块的建立使用了花括号{}。如下例：

```
if ( x>10 ) {
   printf ( "too low, try again\n" ) ;
   scanf ( "%d" , &x) ;
}
```

if语句后和花括号之间的两条语句在x小于10时执行。这两条语句和花括号构成了一个代码块。它们是逻辑单元，必须同时执行。值得注意的是，每条语句既可是单语句，也可是语句块。使用代码块可使算法更清晰、优美和高效，且有助于程序员清楚例程的真实特性。

1.4 C是程序员的语言

令人吃惊的是，不是所有计算机编程语言都针对程序员。例如，BASIC和COBOL就是针对非程序员的。设计COBOL语言并不是为了给程序员带来更多的益处，而是为了改善所产生代码的可读性，甚至也不是为了提高编写代码的速度，而只是部分地使非程序员读懂程序。BASIC语言最初也是为非程序员编写简单的程序并解决简单的问题而设计的。

与之相反，C以及C++完全是由名副其实的程序员所建立、影响和使用的语言。最终的结果是，C语言向程序员提供了所需的一切：限制少，修饰少，块结构，独立函数能力和很少的关键字。通过使用C语言可使程序员达到使用汇编语言的执行效率，同时保留了ALGOL和Modula-2的结构化特性。毫无疑问，在非常优秀的程序员所使用的语言中，C语言是最为通用的编程语言。

C语言得以在程序员之间流行的一个重要因素是它可以代替汇编语言。而汇编语言可以用符号表示计算机所能直接执行的二进制代码。每条汇编语句都对应着计算机实际执行的一个操作。尽管汇编语言可使程序员更为灵活且更富效率地完成其任务，但却很难用汇编语言来研制和调试程序。况且汇编语言是非结构化的，最后编写出的程序一团糟，到处都是转移语句，调用语句和变址语句。这种缺乏结构性的汇编程序让人很难阅读，不易改进和维护。更为重要的是，汇编语言无法在不同的机器（CPU）之间移植。

C语言最初用于系统编程。系统程序是构成部分计算机操作系统和其支持实用程序的大型程序的一部分。下列程序可称为系统程序：

- 操作系统
- 解释程序
- 编辑程序

· 汇编程序
· 编译程序
· 数据库管理程序

随着C语言日益普及，许多程序员利用其可移植性和高效率开始编写各类程序。由于很多机器上都有C语言编译程序，所以可把在一类机器上编写的程序稍加修改或不加修改就能在其它机器上运行。这种可移植性既节省了时间，也节省了费用。此时，C语言编译程序还可以产生紧凑而高效的目标代码。这要比BASIC编程序更佳。

使C语言适用于各种场合的一个重要原因就是程序员们喜欢使用它。C语言拥有汇编语言的速度及FORTH语言的能力，但没有Pascal和Modula-2的限制。每个C程序员都可建立和维护特有的函数库，以适应其自己的风格，并能应用于各种不同的程序之中。鉴于C语言允许并提倡独立编译，程序员可很方便地管理大型项目，并减少不必要的重复性作业。

1.5 C程序的格式

表1.1列出了32个关键字，它们与形式C语法组合形成C编程语言。当然，C语言的原始版本定义了27个关键字。ANSI委员会添加了这样五个关键字：enum, const, signed, void和volatile。

此外，许多C和C++编译程序添加了若干个关键字，便于更好地利用8088/8086系列处理器的内存结构，并支持中间语言编程和中断。最常见的扩展关键字见表1-2。编译程序可能还支持其它扩展，从而有助于更好地利用其稳定环境。

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

表1-1 由ANSI C标准定义的32个关键字

所有C关键字都是小写的。在C中，大小写是不同的：else是关键字，而ELSE却不是。关键字不能在C程序中用于其它目的，也就是说，不能用来作变量或函数的名字。

所有C程序都包含一个或多个函数。唯一不可缺少的函数是main( )函数。在程序开始执行后它第一个被调用。在出色的C程序中，main( )通常含有程序的大致轮廓。该轮廓由一系列函数调用组成。尽管main这个词不是C语言的一部分，但我们常常这样看待它。注意，不要用main作为变量名，否则会给编译程序带来极大混乱。

asm	_cs	_ds	_es
_ss	cdecl	far	huge
interrupt	near	pascal	

表 1-2 一些常见的C/C++扩展关键字

C程序的一般形式见图1-1，其中$f_1(\ )\sim f_N(\ )$表示用户定义函数。

```
global declarations

return-type main(parameter list)
{
  statement sequence
}

return-type f1(parameter list)
{
  statement sequence
}

return-type f2(parameter list)
{
  statement sequence
}

.
.
.

return-type fN(parameter list)
{
  statement sequence
}
```

图 1-1 C程序的基本形式

1.6 库和链接

从技术上讲，可以创建有用的函数型C程序，它们仅由实际建立的语句所组成。不过，这很少见，因为在语言的实际意义内，C并不提供任意执行输入/输出操作的方法。这样，绝大多数程序包含对C标准库中所含各个函数的调用。所有C编译程序都带有一个标准C库。该库提供函数，以执行最常见的任务。ANSI C标准指定包含在库中的最小函数集；不过，你编译程序可能会含有许多其它函数。例如，因为计算环境变化很大，ANSI C标准不定义任意图形函数，但编译程序可能会包含其中几个。

编译程序的实现者们已经编写了绝大多数通用函数。当调用不属于所写程序的函数时，编译程序记得起它的名字。之后，链接程序将所编写的代码与在标准库中已寻找到的目标代码组合。这一进程称为链接（linking）。有些编译程序有自己的链接程序，而其它编译程序使用由操作系统所提供的标准链接程序。

库中的函数格式是浮动的。这就是说，各种机器代码指令的内存地址未绝对定义，仅保留偏置信息。当程序与标准库中的函数链接时，这些内存偏置用于建立所用的实际地址。有一些技术手册和书籍详细解释了这一进程。然而，对用C编程的实际浮动进程的任何进一步解释都是不需要的。

在编写程序时，将会发现在标准库中有许多所需要的函数。这些函数用于建立简单组合