第一部分　C++ 基础:类 C 特性

- C 语言概述
- 表达式
- C 语句
- 数组和串
- 指针
- 函数
- 结构、联合、枚举及用户定义的类型
- 控制台 I/O
- ANSI C 语言标准文件 I/O
- C 语言的预处理程序和注释

本书的第一部分讨论 C++ 语言(以下简称为 C++)的类 C 特性。读者也许知道,C++是 ANSI 标准 C 语言的增强版本。正是基于这一原因,任何 C++编译器都被定义为一个 C 编译器。由于 C++是建立在 C 语言之上的,所以掌握了 C 语言的编程,便能够用 C++编程。C 语言的许多基本概念也是 C++的语言基础,它们将在这部分中一一介绍。由于 C++是 C 语言的扩展,因而本部分的一切描述也都适于 C++。C++的语言特性将在本书的第二部分详述。C++的类 C 特性专门占用一部分内容,这是为了使具有丰富 C 语言编程经验的读者能够方便、迅速地找到 C++的特征信息,而不必重复涉足他们早已掌握的信息。

注意，本书第一部分摘引了《C 语言大全》的部分内容，如果读者对 C 语言特别感兴趣，那么阅读这本书将会很有帮助。它全面介绍了 ANSI C 标准和标准 C 库函数，此外，还有许多 C 语言的应用实例。

第一章　C 语言概述

- C 语言的起源
- C 语言是中级语言
- C 语言是结构化语言
- C 语言是程序员的语言
- C 语言的程序结构
- 库和连接
- 分离编译

本章的目的是介绍 C 语言的概貌、起源、应用情况及构成原理。由于 C++是建立在 C 语言之上的，因此本章也介绍了 C++产生的历史背景。

1.1　C 语言的起源

C 语言是由 Dennis Ritchie 发明并首先在配备 UNIX 操作系统的 DEC PDP-11 计算机上实现的。C 语言是一种比较古老的语言 BCPL 发展过程中的产物。BCPL 是由 Martin Richards 开发的，它影响了由 Ken Thompson 发明的 B 语言，而 B 语言又导致了 C 语言在 70 年代的发展。

多年来，UNIX 操作系统上配备的 C 语言一直被作为 C 语言的公认标准（见 Brian Kernighan 和 Dennis Ritchie 的《程序设计语言 C》，Englewood Cliffs，N. J.：Prentice-Hall，

1978)。随着微型机的发展，出现了一大批C语言系统。由于当时不存在统一的标准，因而不同的实现系统存在着差异和不相容。为了改变这种局面，ANSI在1983年初夏成立了一个委员会以制定一劳永逸的C语言标准。目前，ANSI C标准已被采纳，并且大多数C++和C语言编译器都实现了这个标准。

1.2 C语言是中级语言

C语言通常被称为中级计算机语言。这并不意味着它的功能差，难以使用，或者比BASIC、Pascal那样的高级语言原始，也不意味着它象汇编语言那样，会给使用者带来类似的麻烦。C语言之所以被称为中级语言，是因为它把高级语言的成分同汇编语言的功能结合起来了。下表示出了C语言在计算机中所处的地位。

高级	Ada
	Modula-2
	Pascal
	COBOL
	FORTRAN
	BASIC
中级	C
	FORTH
	Macro-assembler
低级	Assembler

作为中级语言，C支持对位、字节和地址这些有关计算机

功能的基本成分进行操作。C 语言非常容易移植。可移植性表现为可将某种计算机写的软件改编到另一种机器上实现。举一例来说，如果为苹果机写的一个程序能够方便地修改并在 IBM PC 机上运行，则这个程序称为可移植的。

所有的高级语言都支持数据类型的概念。一个数据类型定义了一个变量的取值范围和可在其上操作的一组运算。常见的数据类型是整型、字符型和实型。虽然 C 语言有五种基本数据类型，但与 Pascal 和 Ada 相比，它算不上强类型语言。C 语言支持几乎所有的类型转换。例如，字符型和整型数据能够自由地混合在大多数表达式中。C 语言不支持诸如数组越界等运行错误检查，检查运行错误是程序员的责任。

C 语言的特色是它支持对位、字(字节)和指针的直接操作。这使得它非常适于经常进行上述操作的系统程序设计。C 语言的另一个重要特点是它仅有 32 个关键字(其中 27 个来源于 Kernighan 和 Ritchie 的公认标准，另 5 个是 ANSI 标准化委员会增加的)。这些关键字构成了 C 语言的命令集。和 IBM PC 用的 BASIC 相比，后者包含的关键字多达 159 个。

1.3 C 语言是结构化语言

尽管把 C 语言叫做块结构语言是不严格的，但它还是常被归为结构化语言。因为它与其它结构化语言，如 ALGOL、Pascal 和 Modula-2 有许多相似之处。(从技术上讲，块结构语言允许在过程和函数中定义过程和函数。用这种方法，全局和局部的概念可以通过“作用域”规则加以扩展，“作用域”管理变量的“可见性”。因为 C 语言不允许在函数中定义函数，所以不能称之为通常意义上的块结构语言。)

结构化语言的一个显著特征是程序和数据的分离。这种语言能够把执行某个特殊任务的指令和所有数据信息与程序的其余部分分离开并隐藏起来。获得隔离的一个方法是调用使用局部(临时)变量的子程序。通过使用局部变量,我们能够写出对程序其它部分没有副作用的子程序。这使得编写共享代码段的C程序变得十分简单。如果开发了一些分离得很好的函数,在引用时仅需知道函数做什么,而不必知道它如何去做。切记,过度使用全局变量(可以被全部程序访问的变量)会由于意外的副作用而在程序中引入错误。设计过BASIC程序的人对这个问题都深有体会。

结构化语言为设计者提供了大量的程序设计功能。它直接支持某些循环结构,如WHILE、DO-WHILE和FOR。在结构化语言中禁止或不提倡使用GOTO语句,不能象BASIC和FORTRAN中那样把它作为常用的程序控制语句。结构化语言允许将语句缩行,但又不必拘于严格的程序格式(象FORTRAN语言那种格式)。

下面是一些结构化语言和非结构化语言的例子。

非结构化语言	结构化语言
FORTRAN	Pascal
BASIC	Ada
COBOL	C
	Modula-2

结构化语言是现代化的,而非结构性正是陈旧语言的标志之一。如今,人们普遍认为结构化语言更易于设计和维护。

C语言的主要结构成分是函数——C的独立子程序。在C语言中,函数作为程序构件是一种完成程序功能的基本程序块。函数允许一个程序的不同任务被分别定义和编码,使程

序模块化，具有良好设计风格的函数能保证它正确工作且不会对程序的其它部分产生副作用。能否在大型项目中设计出独立的函数是至关重要的。在这种项目中，一个程序员的代码不应意外地影响其它人的程序。

在C语言中，实现结构化和代码隔离的另一种方法是使用复合语句（或称分程序）。复合语句是程序单位，是一组在逻辑上相互关联的语句。在C语言中，复合语句就是处在左右花括号之间的一串语句。例如：

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

在上例中，如果x小于10，位于if之后、两个花括号之间的两个语句都被执行。这两个句子连同括号表示一个代码块，它们是一个逻辑单位，其中所有语句必须一起执行。注意，每个语句或是一个单语句或是一个复合语句。代码块不仅能使许多算法得到清晰、优美又有效的实现，而且有助于程序员抓住程序的本质。

1.4 C语言是程序员的语言

一看这节的题目，人们可能会产生疑问："所有的计算机语言不都是程序员使用的吗？"回答是否定的。我们看一下典型的非程序设计员的语言COBOL和BASIC。COBOL的设计使程序员难以改善所编写代码的可靠性，甚至不能提高代码的编写速度。然而，COBOL设计者的本意却是打算使非程序人员能读懂程序（这是不大可能的事）。BASIC的主要目的

是允许非专业程序员在计算机上编程解决比较简单的问题。

相反，对于C++和C语言，程序的生成、修改和现场测试自始至终都由真正的程序员进行，其结果是实现了程序员的期望：很少限制、很少强求、块结构、独立的函数以及紧凑的关键字集合。使用C语言编程，既可获得ALGOL或Modula-2的块结构，又具有汇编代码的高效率。这样，C语言很快就成了第一流专业程序员的最常用语言。

C语言被程序员广泛使用的一个原因，实际上是可以用它来代替汇编语言。汇编语言使用的汇编指令，是能够在计算机上直接执行的二进制机器的符号表示。汇编语言的每个操作都映射为计算机执行的单一任务。虽然汇编语言具有使程序员达到最大灵活性和最高效率的潜力，但开发和调试汇编语言程序的困难是难以忍受的。进一步说，由于汇编语言是非结构化的，最后完成的程序肯定象一团意大利面条——纠缠不清的跳转指令、调用指令和下标。非结构性使得汇编语言难于阅读、改进和维护。也许更重要的是，汇编语言程序不能在使用不同的中央处理器的机器之间移植。

最初，C语言被用于系统程序设计。一个“系统程序”是一大类程序的一部分，这类程序构成了计算机操作系统及其实用程序。通常被称为系统程序的有：

操作系统
翻译系统
编辑系统
汇编系统
编译系统
数据库管理程序

随着C语言的普及，许多程序员开始用它设计各类程

序。几乎所有的计算机上都有C语言编译程序，这使我们很少改动甚至不加改动就能将为一种机器写的C语言源程序移植到另一种机器上编译执行。可移植性节省了时间和财力。C编译程序均可产生非常紧凑、执行快捷的目标码。它比所有的BASIC编译程序的目标码更紧凑更快速。

也许C语言被用于所有各类程序设计的主要原因是由于程序员喜欢它。C语言提供了汇编语言的速度和FORTH语言的可扩充性，而很少有Pascal和Modula-2的限制。每个C程序员都可以创建并维护一个专门的函数库。这个库可以根据不同的需要进行裁减，以适应各种程序的设计。C语言支持(更确切地说是鼓励)分别编译，所以可使C程序员方便地管理大型项目，最大限度地减少重复劳动。

1.5 C语言的程序结构

表1-1列举了32个关键字，它们与标准C文法相结合，

表1-1 ANSI C标准定义的32个关键字表

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

形成了程序设计语言C。其中，27个关键字已在C语言的老

版本中被定义。另外5个，即enum、const、signed、void和volatile是ANSI标准委员会新加的。

另外，许多C和C++编译程序都增加了几个关键字以充分利用8088/8086处理机的内存组织，支持中间语言程序设计和中断。常用扩展关键字列在表1-2中。有的编译器也许还支持其它功能，以充分利用其特殊环境。

表1-2 几个通用的C/C++扩展关键字

asm	_cs	_ds	_es
_ss	cdecl	far	huge
interrupt	near	pascal	

C语言的关键字都是小写的。C语言分大写与小写，else是关键字，"ELSE"则不是。在C程序中，关键字不能用于其它目的，即不允许将关键字作为变量名或函数名使用。

所有的C程序都是由一个或多个函数构成的。C语言中至少必须有一个主函数"main()"，它是程序运行开始时被调用的第一个函数。在编制完毕的C代码中，main()实质上包含程序要完成的动作的轮廓。这个轮廓由函数调用组成。虽然从技术上讲，主函数不是C语言的一个保留词，但它仍被这样看待。如"main"用作变量名，就会给编译程序造成混乱。

C语言程序的一般形式如下所示。其中f1()至f2()代表用户定义的函数。本书的其它部分将详细讨论程序的一般形式。

```
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
}
```

1.6 库和链接

从技术上讲，纯粹由程序员自己编写的语句构成C语言程序是可能的，但却不常见，因为C在它的定义中没有执行输入/输出(I/O)操作的任何方法。因此，大多数C程序都含有对C语言的各种标准库函数的调用。所有的C编译程序都是和标准C函数库一起提供的，后者含有完成各种常用任务的函数。ANSI C标准规定了函数库的最小集合，然而，有的编译器也许含有更多的函数。例如，由于计算环境的不同，ANSI C标准并不定义图形函数，但有的编译器也许包含这些函数。

C 编译程序的实现者已经编写了大部分常见的通用函数。但调用一个别人编写的函数时，编译程序要“记住”它的名字。随后，“链接程序”就把编写的程序同标准函数库中找到的目标码结合起来。这个过程称为“链接”。某些编译程序带有自己的链接程序，有些则使用操作系统提供的标准链接程序。

保存在函数库中的函数是可重定位的。这意味着，机器码指令的内存地址并未绝对地确定——只有偏移量是确定的。当把程序与标准函数库中的函数相连接时，内存偏移量被用来产生实际地址。有关的技术手册和参考书中更为详细地讲解了这一过程。然而，用户不必对重定位过程做深入的了解。

编写程序时，用户会在标准函数库中找到许多自己需要的函数。它们都是可以简单地组合起来的程序构件。用户编写的可以重复使用的函数，也可放入库中。有些编译程序允许将用户的函数放入标准库中，有些则允许建立附加库。这两种方法都可将代码放入库中，以便经常使用。

记住，ANSI 标准只描述了最小集的标准库。大多数编译器都支持比 ANSI 定义的通用函数更多的标准库。同样，C 语言 UNIX 版本中的一些函数也未在 ANSI 中定义，因为它们是冗余的。这些函数也许会在用户的编译器中被定义。

1.7 分离编译

大多数 C 和 C++程序都可以包含在一个源文件中。然而，程序越长，编译的时间就越长。因此，C 语言允许用户将一个程序划分成不同块并放入不同文件，分别进行编译。编译好所有文件后，再将它们与库函数一并连接，形成所期望的、完整的执行代码。分离编译的好处是，用户改变一个文件的代码

后，不必重新编译整个程序。这样就节约了大程序的调试时间。一般的C++编译器用户手册都介绍了如何编译复杂程序的指令。

第二章　表达式

- 五种基本数据类型
- 修饰基本类型
- 标识符命名
- 变量
- 存取修饰符
- 存储分类符
- 变量初始化
- 常量
- 运算符
- 表达式

本章介绍C语言的最基本要素——表达式。读者将发现，C语言拥有比其它语言更普遍、更高效的表达式。表达式由C语言的原子要素数据和运算符构成，数据可以是变量或常量。与大多数其它语言一样，C语言支持不同的数据类型。由于数据是一切表达式的核心，因此本章将从C语言的基本数据类型开始介绍。

2.1　五种基本数据类型

C语言有五种基本数据类型：字符、整型、浮点、双精度浮点和无值，分别用char、int、float、double和void来表示。读者

将看到，C 语言中的其它数据类型都是从这些基本类型演变而来的。这些数据的长度和范围因处理器的类型和 C 语言编译程序的实现而异。一般来讲，一个字符为 1 个字节，一个整数为 2 个字节，但不能肯定，用户的程序在移植时不出现字节溢出之类的现象。ANSI C 强调的是每种数据类型的最小范围，而不是字节长度(表 2-1)。

表 2-1　ANSI 标准中的数据类型定义

类型	近似长度(bit)	最小范围
char(字符型)	8	-127～127
unsigned char(无符号字符型)	8	0～255
signed char(有符号字符型)	8	-127～127
int(整型)	16	-32,767～32,767
unsigned int(无符号整型)	16	0～65,535
signed int(有符号整型)	16	同 int
short int(短整型)	16	同 int
unsigned short int(无符号短整型)	16	0～65,535
signed short int(有符号短整型)	16	同 int
long int(长整型)	32	-2,147,483,647～2,147,483,647
signed long int(有符号长整型)	32	同 long int
unsigned long int(无符号长整型)	32	0～4,294,967,295
float(浮点型)	32	约 6 位有效数字
double(双精度型)	64	约 10 位有效数字
long double(长双精度型)	128	约 10 位有效数字

浮点值的准确范围取决于它们是如何实现的。通常，整型对应于宿主机中一个字的长度。从理论上讲，char 类型值受 ASCII 字符所限，但实际上，对 ASCII 字符集范围之外的字

符，C 和 C++都以不同的方式进行了处理。

float 和 double 类型的取值范围通常以数字精度给出。它们的大小由浮点数的表示方法而定。然而，不管用什么表示方法，数值都相当大。ANSI 标准描述浮点值的最小范围是 1e-37～1e+37。

2.2 修饰基本类型

除 void 类型外，基本类型的前面都可以有各种修饰符。修饰符用来改变基本类型的意义，以便更准确地适应各种情况的需求。修饰符有：

signed	（有符号）
unsigned	（无符号）
long	（长型符）
short	（短型符）

这些修饰符适用于字符和整数两种基本类型，其中，long 还适用于 double。

表 2-1 给出了所有根据 ANSI 标准组合的类型、字长和范围。

signed 对整型的修饰是多余的，但仍允许使用，因为整数的缺省定义是有符号数。signed 最重要的用途是用来修饰字符型。

有符号整数和无符号整数的区别是对整数最高位的解释。若指定一个有符号整数，那么 C 编译程序生成代码时将整数最高位作为符号标志。若符号标志是 0，则数值为正；若符号标志是 1，则数值为负。

通常，负数采用 2 的补码形式，即对数的各位（符号标志

除外)求反,若对该数加1,则置符号标志为1。

有符号整数在大量的算法中是很重要的,但其绝对值仅为无符号数值的一半。例如,32767:

01111111 11111111

如果高位设置为1,数值将变为-32767。但是,对于无符号整数,高位置1时,数值将变为65535。

2.3 标识符命名

在C/C++中,标识符是对变量、函数、标号和其它各种用户自定义对象的命名。标识符的长度可以是一个或多个字符。标识符的第一个字符必须是字母或下划线,随后的字符必须是字母、数字或下划线。这里列举一些正确的和错误的标识符命名实例。

正确形式	错误形式
count	lcount
text23	hi! there
high_balance	high..balance

ANSI C标准规定,标识符可以为任意长度。但是,若标识符用于链接过程中,则必须至少能由前6个字符唯一地区分。这些标识符称为外部名,包括文件间共享的函数名和全局变量名。若标识符不用于链接过程中,则必须至少能由前31个字符唯一地区分,这些标识符称为内部名。然而,在C++中,标识符没有长度限制,所有字符都是可区分的。这种差异在C程序转换成C++程序的过程中变得十分重要。

C语言中的字母是有大小写之分的,因此count、Count、COUNT代表不同的标识符。不过,如果链接程序中不区分大