



普通高等教育“十二五”规划教材

C语言程序设计教程

主 编 李刚健 段 淼 郑 琦
副主编 刘舒婷 刘立辉 王祥瑞 陈 静 崔立波
主 审 李 杰



中国水利水电出版社
www.waterpub.com.cn

普通高等教育“十二五”规划教材

C 语言程序设计教程

主 编 李刚健 段 淼 郑 琦

副主编 刘舒婷 刘立辉 王祥瑞 陈 静 崔立波

主 审 李 杰



中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书通过对教学内容的基础性、科学性和前瞻性的研究,提炼科学内涵,体现以基本理论为主体,反映计算机科学领域的最新科技成果,全面系统地介绍了C语言程序设计的思想、方法和技巧。

全书共分为13章,具体介绍C语言程序的数据类型、数据运算、输入/输出函数、结构化程序设计、数组、函数、指针、结构体、共用体、枚举类型、文件系统以及位运算等内容,并在附录中增加了计算机公共基础二级考试要点(数据结构、软件工程、数据库技术、程序设计基础)四个方面的知识,这四方面的内容在计算机专业中是四门非常重要的专业课,对于从事软件开发的人员来说,是必须要掌握的内容。教材具有较大的知识信息量,从程序设计的基础知识、流程控制到最后的文件操作,由浅至深,层层引导,让学生能够快速掌握C语言,提高编程能力。

本书内容丰富、结构清晰、图文并茂,易于教学与自学,可以作为高等学校相关专业本、专科的学生学习程序设计类课程的教材使用,也可供程序设计爱好者自学和参考使用。

图书在版编目(CIP)数据

C语言程序设计教程 / 李刚健, 段淼, 郑琦主编. —
北京: 中国水利水电出版社, 2014. 5
普通高等教育“十二五”规划教材
ISBN 978-7-5170-1970-1

I. ①C… II. ①李… ②段… ③郑… III. ①C语言—
程序设计—高等学校—教材 IV. ①TP312

中国版本图书馆CIP数据核字(2014)第090384号

策划编辑: 崔新勃

责任编辑: 陈 洁

封面设计: 李 佳

书 名	普通高等教育“十二五”规划教材 C语言程序设计教程
作 者	主 编 李刚健 段 淼 郑 琦 副主编 刘舒婷 刘立辉 王祥瑞 陈 静 崔立波 主 审 李 杰
出版发行	中国水利水电出版社 (北京市海淀区玉渊潭南路1号D座 100038) 网址: www.waterpub.com.cn E-mail: mchannel@263.net (万水) sales@waterpub.com.cn
经 售	电话: (010) 68367658 (发行部)、82562819 (万水) 北京科水图书销售中心(零售) 电话: (010) 88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排 版	北京万水电子信息有限公司
印 刷	北京蓝空印刷厂
规 格	184mm×260mm 16开本 16印张 405千字
版 次	2014年5月第1版 2014年5月第1次印刷
印 数	0001—5000册
定 价	35.00元

凡购买我社图书,如有缺页、倒页、脱页的,本社发行部负责调换

版权所有·侵权必究

前 言

社会信息化不断向纵深发展，各行各业的信息化进程不断加速。计算机应用技术与其他专业的教学、科研工作的结合更加紧密。各学科与以计算机技术为核心的信息技术的融合，促进了计算机学科的发展，各专业对学生的计算机应用能力也有更高和更加深入的要求。

基于近年来计算机科学的发展以及国家教育部关于计算机基础教学改革和指导思路，我们确立了本套教材的编写思想。本书的所有编者均为一线教师，根据长期 C 语言程序设计教学中所遇到的问题有针对性地编写了此教材。在本教材的策划过程中，教材编委会组织召开了多次研讨会，对现有比较出色的教材的特点进行了分析，博采众长，力求实现教材权威性与实用性的完美结合。

另外，我们在国内高校做了系统、详细的调研，对教育部制订的教育计划做了认真的研究，还对国内外已出版的教材做了理性的分析，确立了依托国家教育计划、传播先进教学理念、为培养符合社会需要的高素质创新型应用型人才服务的创作宗旨。

本书详细而全面地介绍了 C 语言程序设计的思想、方法和技巧。全书共 13 章，主要内容具体包括：C 语言概述、数据类型、数据运算、输入/输出函数、顺序结构、选择结构、循环结构、数组、函数、指针、编译预处理、结构体、共用体、枚举类型、文件操作以及位运算等，附录中收录了计算机公共基础二级考试要点（数据结构、软件工程、数据库技术、程序设计基础）四个方面的知识。全书内容丰富，覆盖面广，立意新颖，讲述深入浅出、结构合理、实践性强，既注重理论又突出使用性，使学生较全面快速地掌握 C 语言程序的基本概念、基本原理、技术与方法，提高程序设计能力和水平。

本书可作为高等学校相关专业本、专科学生学习程序设计类课程的教材使用，也可供程序设计爱好者自学和参考使用。

教育的改革不会停止，教材也将不断地推陈出新。由于时间仓促和作者水平有限，书中难免有疏漏和不妥之处，欢迎使用本套教材的学校师生和广大读者不吝批评指正，以便改进我们的工作，使教材质量不断提高。

编 者
2014 年 1 月

目 录

前言

第1章 C语言概述	1	3.4.1 赋值运算符	27
1.1 C语言的发展与特点	1	3.4.2 复合赋值运算符	28
1.1.1 C语言的发展	1	3.5 关系运算	28
1.1.2 C语言的特点	1	3.6 逻辑运算	29
1.2 C语言的符号系统	2	3.7 逗号运算	30
1.3 C程序的基本结构	3	习题3	31
习题1	7	第4章 顺序结构程序设计	32
第2章 C语言数据类型	8	4.1 结构化程序设计基础	32
2.1 数据类型	8	4.2 C程序中的语句类型	33
2.1.1 数据类型种类	8	4.3 字符输入/输出函数	36
2.1.2 常量与变量	8	4.3.1 字符输出函数	36
2.2 整型数据	10	4.3.2 字符输入函数	37
2.2.1 整型常量	10	4.4 格式输入/输出函数	38
2.2.2 整型变量	10	4.4.1 格式输出函数	38
2.3 实型数据	12	4.4.2 格式输入函数	42
2.3.1 实型常量	12	4.5 程序举例	45
2.3.2 实型变量	13	习题4	48
2.4 字符型数据	14	第5章 选择结构程序设计	50
2.4.1 字符常量	14	5.1 if结构	50
2.4.2 字符变量	15	5.1.1 if语句	50
2.4.3 字符串常量	16	5.1.2 if_else语句	51
2.5 库函数的使用	16	5.1.3 if语句的嵌套	53
习题2	17	5.2 switch结构	58
第3章 C语言数据运算	18	5.2.1 switch语句	58
3.1 数据运算	18	5.2.2 switch语句中的break语句的使用	60
3.1.1 运算符	18	5.3 程序举例	64
3.1.2 优先级及结合性	19	习题5	73
3.2 算术运算	20	第6章 循环结构程序设计	74
3.2.1 算术运算符	20	6.1 while语句	74
3.2.2 算术表达式	21	6.2 do...while语句	76
3.2.3 自增自减运算符	22	6.3 for语句	78
3.3 混合运算与类型转换	24	6.4 循环结构中的跳转语句	81
3.4 赋值运算	27	6.4.1 break语句	81

6.4.2 continue 语句	82	8.3.5 字符串处理函数	124
6.5 循环的嵌套	83	8.4 数组作为函数参数	127
6.6 程序举例	84	8.4.1 数组元素作为函数参数	127
习题 6	86	8.4.2 数组名作为函数的参数	128
第 7 章 函数	87	8.5 程序举例	130
7.1 函数的定义	87	习题 8	133
7.1.1 函数的定义形式	87	第 9 章 指针	135
7.1.2 函数的使用说明	88	9.1 指针概述	135
7.2 函数参数与返回值	89	9.1.1 指针的概念	135
7.2.1 形式参数和实际参数	89	9.1.2 指针变量的定义与初始化	136
7.2.2 函数的返回值	92	9.1.3 指针运算	138
7.3 函数的调用	92	9.1.4 多级指针	141
7.3.1 函数调用的方式	92	9.2 指针与函数	143
7.3.2 对被调用函数的声明	94	9.2.1 指针作为函数参数	143
7.3.3 函数的嵌套调用	97	9.2.2 指针函数	145
7.3.4 函数的递归调用	98	9.2.3 指向函数的指针	146
7.4 变量的作用域	102	9.3 指针与数组	149
7.4.1 局部变量	102	9.3.1 指向一维数组的指针	149
7.4.2 全局变量	103	9.3.2 二维数组与多维数组的指针表示	152
7.5 变量的存储类型	105	9.4 指针与字符串	157
7.5.1 变量的动态与静态存储方式	105	9.4.1 字符串的指针表示	157
7.5.2 局部变量的存储类型	106	9.4.2 字符串数组	159
7.5.3 全局变量的存储类型	107	9.5 指针数组与命令行参数	160
7.6 内部函数与外部函数	109	9.5.1 指针数组	160
习题 7	109	9.5.2 指针数组与命令行参数	161
第 8 章 数组	110	9.6 程序举例	163
8.1 一维数组	110	习题 9	167
8.1.1 一维数组的定义	110	第 10 章 编译预处理	169
8.1.2 一维数组的初始化	110	10.1 宏定义#define	169
8.1.3 一维数组的使用	111	10.1.1 无参宏定义	169
8.2 二维数组	113	10.1.2 有参宏定义	170
8.2.1 二维数组的定义	114	10.2 文件包含#include	172
8.2.2 二维数组的初始化	115	10.3 条件编译	173
8.2.3 二维数组的使用	116	习题 10	174
8.3 字符数组与字符串	118	第 11 章 结构体、共用体与枚举类型	175
8.3.1 字符数组及其初始化	118	11.1 结构体类型的声明	175
8.3.2 字符串的输入	121	11.2 结构体变量	175
8.3.3 字符串的输出	122	11.2.1 结构体变量的定义与存储	176
8.3.4 二维字符数组	122	11.2.2 结构体变量的引用与初始化	177

11.3 结构体数组	178	12.5 出错检测	212
11.4 结构体指针	180	习题 12	213
11.5 结构体嵌套	181	第 13 章 位运算	214
11.6 动态存储分配	182	13.1 位运算符和位运算	214
11.7 链表	183	13.1.1 位运算符	214
11.7.1 链表的基本概念	183	13.1.2 按位与运算	214
11.7.2 链表的基本操作	184	13.1.3 按位或运算	216
11.8 共用体	189	13.1.4 按位异或运算	217
11.9 枚举类型	191	13.1.5 按位求反运算	219
11.9.1 枚举类型的声明与定义	192	13.1.6 左移运算	220
11.9.2 枚举类型的使用方法	193	13.1.7 右移运算	220
11.10 类型名重新定义 typedef	194	13.1.8 不同长度的数据进行位运算	221
习题 11	195	13.2 位段结构	221
第 12 章 文件操作	197	13.2.1 位段的概念	221
12.1 文件系统的概念	197	13.2.2 位段结构的定义和位段变量 的说明	221
12.1.1 文件输入/输出的基本概念	197	13.2.3 位段的引用	223
12.1.2 文件的分类	198	13.2.4 位段的赋值	223
12.2 文件类型指针	199	13.2.5 使用位段的注意事项	223
12.3 文件操作	200	习题 13	224
12.3.1 建立数据文件	200	附录 A 计算机公共基础二级考试要点	225
12.3.2 文件指针变量	200	附录 B ASCII 码表	243
12.3.3 文件的打开	201	附录 C C 语言库函数	244
12.3.4 文件的关闭	202	参考文献	249
12.3.5 文件的读写	202		
12.4 文件的随机读写	210		

第 1 章 C 语言概述

C 语言是目前广泛应用的一种计算机程序设计语言，它作为一种高级的程序设计语言，具备方便性、灵活性和通用性等特点。同时，它还向程序员提供了直接操作计算机硬件的功能，具备低级语言的特点，适合各种类型的软件开发。因此，C 语言是深受软件工作者欢迎的程序设计语言。

本章首先概要介绍了 C 语言的发展和特点，然后着重讨论了 C 语言的字符集与标识符，最后通过几个简单的程序实例描述了 C 语言程序的基本结构。

1.1 C 语言的发展与特点

1.1.1 C 语言的发展

C 语言是由美国 Bell 实验室的 Dennis Ritchie 设计，在 BCPL 语言和 B 语言的基础上，于 1972 年在一台使用 UNIX 操作系统的 DEC PDP-11 计算机上实现的，它取 BCPL 的第二个字母而得名。

BCPL (Basic Combined Programming Language) 语言是由英国剑桥大学的 Martin Richards 于 1967 年提出的一种系统程序语言。最初的 C 语言主要是在美国 Bell 实验室内部使用，随着 UNIX 操作系统被日益广泛的使用，C 语言也迅速得到推广。在 C 语言与 UNIX 操作系统的发展过程中，它们相辅相成，相互促进。

1983 年初夏，美国国家标准化协会 (ANSI) 专门成立了一个委员会，为 C 语言制定了 ANSI 标准，称为 ANSI C，比原来的 C 有了很大的发展。Turbo C、Visual C++ 等完全是按照 ANSI 的 C 语言标准实施的，是一种快速、高效的编译程序，并提供了集成开发环境。

C 语言不仅可用来开发系统软件，也可用来开发应用软件。C 语言具有一般高级语言的特性（可读性、可移植性好，结构化程序设计），同时具有低级语言（如汇编语言）的特性（可直接对硬件进行操作，如对位、字节和内存地址的操作），它是一种集高级语言和低级语言优点于一身的语言。因此，C 语言常被称为计算机“中级语言”。

1.1.2 C 语言的特点

C 语言能得到越来越广泛的使用，归功于它有不同于（或优于）其他语言的特点，特别是优良的“可移植性”。概括起来，C 语言的主要特点如下：

(1) 结构简洁、紧凑。C 语言一共只有 32 个关键字（即保留字），压缩了一切不必要的成分，因此 C 语言编写的源程序短，但 C 语言却被公认为是一门比较强有力的语言。另一方面，C 语言的运算符和表达式的表示方法也力求简练，易于学习，便于理解和使用。

(2) 数据类型丰富，控制结构完善。C 语言数据类型丰富，它不仅有不同长度的整型、字符型和单双精度实型（即浮点型）等基本数据类型，还有指针类型以及数组、结构体和联合体等构造数据类型，能用这些类型来实现各种复杂数据结构（如链表、树、栈等）的运算。

C 语言提供了 9 种控制语句来实现 3 种结构（顺序、分支和循环结构）的程序设计，以支持良好的程序结构。同时用函数作为程序模块以实现程序的模块化，每一个 C 程序由若干个函数组成，用户不仅可以调用丰富的库函数，还可以自定义函数。因此，C 语言是结构化的理想语言，符合现代编程风格要求。

(3) 运算符丰富，表达能力强。C 语言的运算符非常丰富，共有 34 种运算符（详见第 3 章）。它不仅具有算术运算符、关系运算符、逻辑运算符等，还把括号、赋值、逗号、变量值增（减）1、强制类型转换等都作为运算符处理，同时还将字符、逻辑值等数值化。C 语言的运算符不仅具有优先级的概念，还具有结合性的概念。灵活使用各种运算符和表达式，不仅可以使程序简洁，还可以实现在其他语言中难以实现的运算。

(4) 语法限制不太严格，程序设计自由度大。C 程序的书写形式自由，主要用小写字母，一个语句可以写在几行，一行也可以写几个语句。

一般的高级语言语法检查比较严，能检查出几乎所有的语法错误，而 C 语言允许编写者有较大的自由度，因此放宽了语法检查。程序员应仔细检查程序，保证其正确，而不要过分依赖 C 编译程序去查错。

(5) 同时具有高级语言和低级语言的特点。C 语言允许直接访问物理地址，能进行位操作，可以直接对硬件进行操作，能实现汇编语言的大部分功能。因此，C 语言既具有高级语言的功能，又具有低级语言的许多功能。

(6) 目标代码质量高，可移植性好。C 语言程序不仅编译的目标代码质量高，程序执行效率高，而且可移植性好。虽然 C 语言适合用在许多计算机机型上，它却独立于具体的计算机系统，灵活地运用预处理程序可以提高程序的可移植性。

1.2 C 语言的符号系统

1. 字符集

满足 C 语言词法要求的字符包括所有大小写的英文字母、阿拉伯数字以及部分特殊符号，这些特殊符号如图 1.1 所示。

+	-	*	/	%	_ (下划线)	=	<	>	&
~	(	)	[	]	.	{	}	:	?
;	"	!	#	'	^	空格			

图 1.1 C 语言字符集中的特殊符号

2. 标识符

C 语言的标识符（identifier）主要用来表示常量、变量、函数和类型等的名字，它们是只起标识作用的一类符号，包括保留字、预定义标识符和用户定义标识符三类。

保留字是指具有特定含义的字符串，这些字符串不允许用户把它们当做变量名字使用。C 语言的保留字共有 32 个，它们一般都用小写英文字母表示，如图 1.2 所示，它们的含义在后面各章中将会逐步介绍。

除了上述的保留字之外，还有一类具有特殊含义的标识符，它们被用作库函数名和预编译命令，这类标识符在 C 中称之为预定义标识符。一般来说，也不要把这些标识符再定义为其其他标识符使用。

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

图 1.2 C 语言的保留字

预定义标识符包括预编译命令和 C 编译系统提供的库函数名。常见的预编译命令有 `define`、`include`、`ifdef`、`ifndef`、`endif`、`elif` 等。

用户定义标识符是程序员根据自己的需要定义的一类标识符，用于标识变量、符号常量、用户定义函数、构造类型名和文件名等。这类标识符由英文字母、数字和下划线构成，但开头字符只能是字母或下划线。下划线常用来把几个词语隔开，以增强代码的可读性，如变量 `num_of_people`。

C 语言对大小写敏感，保留字和预定义标识符全部使用小写字母，用户定义标识符也需要区分大小写，如 `NUMBER`、`Number` 和 `number` 被视为不同的标识符。

一个用户定义标识符可由许多字符组成，但其长度是有限制的，ANSI C 只能识别前 31 个字符，因此如果两个标识符的前 31 个字符是相同的，那么它们将被视为同一个标识符。

1.3 C 程序的基本结构

下面介绍几个简单的 C 程序，并分析 C 程序的构成及其特性。

例 1.1 求两个数之和。

```
#include<stdio.h>          /*将标准输入输出函数的头文件包含到本文件中来*/
void main()                /*主函数头*/
{
    int a,b,sum;            /*定义 a、b、sum 为整型变量*/
    a=1897; b=100;         /*给变量 a、b 赋值*/
    sum=a+b;               /*给 sum 赋值*/
    printf("a=%d, b=%d\n", a, b); /*输出变量 a、b 的值*/
    printf("sum is %d\n", sum);  /*输出变量 sum 的值*/
}
```

程序运行结果：

```
a=1897, b=100
sum is 1997
```

本程序的作用是求两个整数 `a` 和 `b` 之和，并将和值存放在变量 `sum` 中。

`main` 是主函数名，每一个 C 程序都必须有一个 `main` 函数；函数名后的圆括号内是函数的参数，本例是一个无参函数。函数体是由花括号 `{}` 括起来的部分，函数体内首先是变量的定义（也称为变量的声明）部分，然后是执行语句部分。C 语言的每个变量定义语句和每一条执行语句都是以分号“`;`”结束。

第四行定义变量 `a`、`b` 和 `sum` 是整型（`int`）变量。第五行和第六行均为赋值语句。第五行包含两条赋值语句，分别给变量 `a` 和 `b` 赋值为 1897 和 100；第六行是将表达式 `a+b` 的值赋给变量 `sum`。

第七行和第八行均是输出函数调用语句，`printf` 是 C 语言中的输出函数（详见第 4 章）。

双引号引起来的是格式控制字符串，它可以包括“格式说明”和普通字符，其中每一个“格式说明”都是由“%”接格式字符构成，字符“d”是以十进制形式输出整型数据的格式字符；格式控制字符串中的普通字符照原样输出，其中“\n”是转义字符，表示换行（即换行符）。printf 函数中括弧内最右边给出输出表达式，这里给出的是变量 a、b 或 sum，输出表达式的个数应该与格式控制字符串中“格式说明”的个数一致。

“/*.....*/”是注释部分，它表示从“/*”开始，一直到“*/”之间的内容为注释，注释只是给人看的，对编译和运行不起作用。注释可以加在程序中任何位置。对于一行内的注释，也可以使用“//.....”的形式，它表示本行中从“//”开始的内容为注释。

第一行是编译预处理命令中的“文件包含”命令（详见第 10 章），它的作用是将标准输入输出函数的头文件 stdio.h 包含到本程序文件中来。stdio.h 头文件中包含了标准输入输出操作有关的函数的原型声明等信息，因此，凡是需要使用标准输入输出函数（如 printf、scanf 等函数）的 C 程序都必须在文件的开头加上这条文件包含命令，以便编译系统在编译的时候能够从头文件中获得有关的信息。

例 1.2 从键盘输入一个整数，判断它是负数还是非负数。

```
#include<stdio.h>
void main()
{
    int x;
    printf("Please input an integer number: \n");
    scanf("%d", &x);
    if (x<0)printf("It is a negative.\n");
    else printf("It is not a negative.\n");
}
```

第一次运行情况如下：

```
Please input an integer number:
-5↵
It is a negative.
```

第二次运行情况如下：

```
8↵
It is not a negative.
```

说明：在本书中，↵代表回车键。

第五行输出函数调用语句是用来输出一行提示信息，双引号内的字符串原样输出。第六行是输入函数调用语句，scanf 是 C 语言中的输入函数（详见第 4 章）。双引号引起来的是格式控制字符串，含义同前例。scanf 函数中括弧内最右边给出输入变量存储单元指针（请读者先将指针的概念理解为存储单元的地址，有关指针的概念详见第 9 章），这里给出的是变量 x 的存储单元地址，“&”是求地址运算符。输入变量存储单元地址的个数应该与格式控制字符串中“格式说明”的个数一致。如果格式控制字符串中有普通字符，则在输入时这些普通字符也得照原样输入。

第七行及第八行是分支选择语句，它的作用是：如果变量 x 是负数（即 $x < 0$ 成立），则输出 "It is a negative."；否则（即 $x < 0$ 不成立），则输出 "It is not a negative."。

例 1.3 求 1 至 100 之间的所有自然数之和。

```
#include<stdio.h>
void main()
```

```

{
    int sum=0, i=1;          /*定义 sum 和 i 为整型变量，并分别赋值为 0 和 1*/
    while(i<=100)           /*如果 i<=100 成立，则重复执行循环体语句*/
    {
        sum=sum+i;          /*循环体语句是由两条语句构成的复合语句*/
        i=i+1;
    }
    printf("sum is %d\n", sum);
}

```

程序运行结果：

sum is 5050

第四行定义变量 `sum` 和 `i` 为整型变量，同时分别给它们赋值为 0 和 1。在变量定义语句中给变量赋值称为变量的初始化。

第五至第九行是一个循环语句，它的作用是：若条件成立（这里的条件是“`i<=100`”），则重复执行循环体语句。这里循环体语句是由“`sum=sum+i;`”和“`i=i+1;`”两条语句组成。C 语言规定循环体语句只能是一条语句，因此需要将这两条语句组合成一条复合语句来使用。用一对花括号 `{}` 将若干条语句括起来可以构成一条复合语句。

例 1.4 从键盘输入三个整数，求出它们之中最大的数。

```

#include<stdio.h>
int max(int x, int y)
{
    /*定义 max 函数，返回值为整型，x、y 为形式参数，x、y 为整型*/
    int z;                          /*定义 max 函数中用到的变量 z 的类型*/
    if(x>y) z=x; else z=y;          /*分支循环语句*/
    return (z);                     /*返回变量 z 的值，通过 max 带回到调用处*/
}
void main()
{
    int maxnum, a, b, c;
    printf("Please input 3 integer numbers: \n");
    scanf("%d, %d, %d", &a, &b, &c); /*从键盘输入三个整数分别 a、b、c*/
    maxnum=max(a, b);                /*将 a 和 b 中较大的值赋值给变量 maxnum*/
    maxnum=max(maxnum, c);           /*将 maxnum 和 c 中较大的值赋值给 maxnum*/
    printf("Max number is %d\n", maxnum); /*输出变量 maxnum 的值*/
}

```

程序运行结果：

```

Please input 3 integer numbers:
5, 3, 8
Max number is 8

```

本程序由两个函数构成：主函数（即 `main` 函数）及用户自定义函数 `max`。函数 `max` 的作用：将形参变量 `x`、`y` 中较大者的值赋值给变量 `z`，并通过 `return` 语句将变量 `z` 的值返回给主调函数。形参变量的值是从主调函数中的函数调用语句的实参中传送过来。`main` 函数的作用：从键盘输入三个整数分别给变量 `a`、`b`、`c`，通过调用函数 `max` 求出三个数中最大一个的值。调用函数 `max` 一次可以求出两个数中较大的一个，现要求求出三个数中最大的一个值，

因此需要两次调用函数 `max`。

`main` 函数中第一次调用 `max` 函数时，将实参 `a`、`b` 的值分别传送给 `max` 函数中的形参变量 `x`、`y`；通过执行 `max` 函数得到一个返回值（即 `max` 函数中变量 `z` 的值），把这个值（它是变量 `a` 和 `b` 中较大一个的值）赋值给变量 `maxnum`。第二次调用 `max` 函数时，将实参 `maxnum`、`c` 的值分别传送给 `max` 函数中的形参变量 `x`、`y`；通过执行 `max` 函数得到一个返回值，把这个值（它是变量 `maxnum` 和 `c` 中较大一个的值）再赋给变量 `maxnum`。两次调用函数 `max` 后，变量 `maxnum` 已经是变量 `a`、`b`、`c` 中最大一个的值。

通过以上几个例子，可以看到：

（1）C 程序是由函数构成的。一个 C 程序必须包含且只能包含一个 `main` 函数，不能有一个以上的 `main` 函数；除 `main` 函数外，可以包含若干个其他函数。被调用的函数可以是库函数（即系统提供的标准函数），也可以是用户根据需要编写的函数（即用户定义的函数）。函数是 C 程序的基本组成单位。

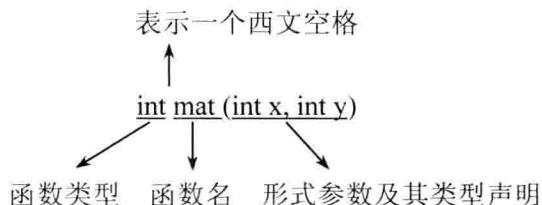
C 语言的函数相当于其他语言的子程序、过程。C 语言的这种特点使其容易实现程序的模块化。

（2）一个 C 函数由函数头和函数体两部分构成。详见第 7 章。

1）函数头部分。包括函数属性（如内部函数和外部函数）、函数类型（即函数返回值的类型）、函数名、形式参数及其类型声明列表。

一个函数名后面必须跟一对圆括弧，函数参数可以没有，这对圆括弧不能没有。

例如，例 1.4 中的 `max` 函数的头部分为：



2）函数体。即函数头部分以下的花括号 `{}` 内部的部分。如果一个函数内有多个花括号，则最外层的一对 `{}` 为函数体的范围；内层 `{}` 为复合语句。

函数体一般包括：

①变量定义部分。变量定义部分一般应放在函数体的执行语句之前，一个函数可以没有变量定义部分。

②执行语句部分。由若干个语句组成，一个函数可以没有执行语句部分。如空函数：

```
void dump()
{
}
```

既没有变量定义部分，又没有执行语句部分，它什么也不干，但这是合法的。

（3）一个 C 程序中各个函数的定义是独立的、平行的。无论各个函数之间的位置如何安排，一个 C 程序总是从 `main` 函数开始执行；如果没有出现意外的话，它也总是从 `main` 函数结束整个程序的运行。

（4）C 程序书写格式自由，一行内可以写几个语句，一个语句也可以写在几行上。C 程序的语句没有行号，只要求每一个变量声明和执行语句均以分号“`;`”作为结束，分号是 C 语

句的必要组成部分。

(5) C语言本身没有输入输出语句。输入输出的操作是由库函数 `scanf` 和 `printf` 等来完成的。C语言对输入输出实行“函数化”。

(6) 可以用 `/*.....*/` 对 C 程序中的任何部分作注释。为增加程序的可读性,源程序都应加上必要的注释。

习题 1

1.1 指出下面哪些是合法的标识符,其中哪些是关键字,哪些是用户定义标识符。

Abc	2x	student	v1r	m+y	a#
INT	until	let	b-4	else	_rt
const	name	x%y	do	w tm	case

1.2 什么是算法?请用描述“输出三个数中最小数”的算法。

1.3 结构化程序的三种基本结构是什么?C程序中的函数由哪两部分构成?

第2章 C语言数据类型

C语言可以处理数值型及非数值型数据。数据类型用于描述数据的特征，决定了数据被存储时所占内存字节数、数据取值范围和其可进行的操作。数据类型越丰富，功能越强，可以编写越复杂的应用程序。本章主要介绍C语言的三种基本数据类型，即整型、实型和字符型，每种数据类型又有常量和变量之分。

2.1 数据类型

2.1.1 数据类型种类

C语言有丰富的数据类型，它既有基本类型，又有构造类型，还有指针类型和空类型。C语言的数据类型如图2.1所示。

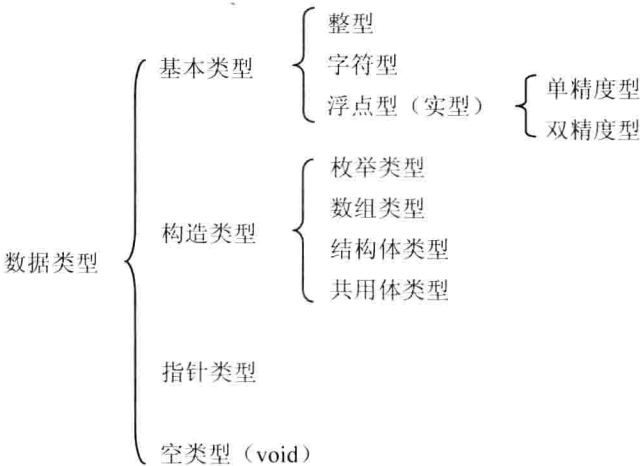


图 2.1 C语言的数据类型

2.1.2 常量与变量

常量是指在程序运行过程中，其值保持不变的量。在C语言中，有整型常量、实型常量、字符常量、字符串常量和符号常量等类型。

变量是指用于存储数据，程序运行中其值可以被改变。每一个变量都必须有一个名字，即变量名。程序中定义了一个变量，即表示在内存中拥有了一个可供使用的存储单元，用来存放数据，即变量的值，而变量名则是编程者给该存储单元所起的名称。

程序运行过程中，变量的值存储在内存中。从变量中取值，实际上是根据变量名找到相应的内存地址，从该存储单元中读取数据。在定义变量时，变量的类型必须与其被存储的数据类型相匹配，以保证程序中变量能够被正确的使用。当指定了变量的数据类型时，系统将为其

分配若干相应字节的内存空间。如 `char` 型为 1 个字节，`int` 型为 4 个字节，`float` 为 4 个字节，`double` 为 8 个字节。

在 C 语言中对变量、符号常量、函数等数据对象命名的有效字符序列统称为标识符。变量的命名必须遵循标识符的命名规则。变量命名时，应该做到“见名知意”。通常选择能表示数据含义的英文单词或缩写作为变量名，以提高程序的可读性。例如，`name` 表示姓名，`sex` 表示性别，`age` 表示年龄。

C 语言规定变量必须先定义，后使用，未经定义的变量不能使用。被定义的变量在没有赋初始值之前，其值是不确定的，因而必须指定变量的类型及初始值以便进行下一步运算。

变量赋初值的格式：**变量=表达式；**

例如：

```
int a;
a=9;
```

C 语言规定，可以在定义变量的同时给变量赋初值，即变量初始化。变量初始化只需定义变量时在变量名后面加一个赋值号和一个常数。

例如：

```
int a=9;
float b=7.89;
char c='x';
```

相当于：

```
int a;
float b;
char c;
a=9;b=7.89;c='x';
```

变量赋值的原则是变量与常量的数据类型应一致或相容。给变量赋值时，可以给被定义的变量的一部分赋初值。

例如：

```
int a,b,c=6;
```

相当于：

```
int a,b,c;
c=6;
```

也可以给几个变量赋同一个初值。

例如：

```
int a=b=c=6;
```

相当于：

```
int a,b,c;
a=6;b=6;c=6;
```

不同类型的数据在编译时分配的内存空间大小不同，例如字符型占 1 个字节，整型（`int`）占 4 个字节，而浮点型（`float`）占 4 个字节。

不同类型的数据在内存中的存储形式不同，例如，字符型是以 ASCII 码值存储的，整型是以补码的形式存储的，浮点型是以指数的形式存储的。

不同类型的数据所使用的运算符不同，这就便于在编译时据此检查在程序中要求对该变量进行的运算是否合法。

请思考：C 语言中变量名与其类型有关吗？为什么变量必须先定义后使用？

2.2 整型数据

2.2.1 整型常量

整型常量又称为整常数，可以用十进制、八进制、十六进制表示。

十进制整数是由正负号和 0~9 之间的数字组成的数据，并且第一位数字不能是 0。例如 2304，-129 都是合法的十进制整数。

八进制整数是由正负号和 0~7 之间的数字组成的数据，例如，0123 即十进制数 83；-012 即十进制数-10；0177777 即十进制数 65535。

十六进制整数由 0x（或 0X）开头，由数字 0~9、英文字母 a~f（或 A~F）组成的数据，其中 a~f（或 A~F）表示十进制数 10~15。例如，0x123 即十进制数 291；-0x12 即十进制数-18；0xFFFF 即十进制数 65535。

整常数在不加特别说明时总是正值。如果是负值，则符号“-”必须放在常数的前面。数值超过整型数据范围时，可使用长整型数，每个长整型数占 4 个字节（32 位）存储空间。长整型常量的表示形式是在整型常量的后面跟一个字母 L 或 l，即为 long int 型。例如 123L、1234567L。

在一个整型常量后面跟一个字母 u 或 U 为无符号整型常量，即 unsigned int 型，例如 12345u。

无符号整型常量在内存中存放时最高位不作为符号位，而是用来存储数据。

请思考：16L 与 16 有什么区别？

2.2.2 整型变量

除了基本整型外，还可以在 int 前加上修饰符来改变整型的意义，修饰符有 signed（有符号）、unsigned（无符号）、long（长型）、short（短型）。

整型变量的说明加上修饰符后，int 可以省略。整型变量加上修饰符后，其取值范围有所变化。以 32 位机为例，各种形式整型变量的取值范围如表 2.1 所示。

表 2.1 整型变量的取值范围

名称	类型说明符	位数	取值范围
基本整型	int	32	-2147483648~2147483647
无符号基本整型	unsigned int	32	0~4294967295
短整型	short [int]	16	-32768~32767
无符号短整型	unsigned short [int]	16	0~65535
长整型	long [int]	32	-2147483648~2147483647
无符号长整型	unsigned long [int]	32	0~4294967295

方括号内的部分是可以省略不写的。例如 unsigned short int 与 unsigned short 是等价的。有符号整型数据的最高位为符号位。0 表示正，1 表示负。有符号整数对于许多运算都是