

21世纪高等学校教材

C语言程序设计教程

第二版

主编 李明 副主编 宣善立



上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

21 世纪高等学校教材

C 语言程序设计教程

(第二版)

主 编 李 明

副主编 宣善立

上海交通大学出版社

内 容 提 要

C 语言是当今软件开发领域中广泛应用的一种语言,也是高等学校计算机语言类课程的首选语言。本书共分为 10 章,系统全面地介绍了 C 语言的基本概念、C 语言的数据类型、C 语言的程序构成,系统阐述了各种程序设计的方法。本书内容合理、案例丰富,讲解深入浅出、循序渐进,既注重培养学习者程序设计的能力,又提倡良好的程序设计风格。

本书可作为 C 语言程序设计课程的教学用书,也可作为学习 C 语言的自学教材。为配合本书的学习,本书配有 CAI 教学课件,并有配套的《C 语言程序设计实验指导与习题解答》,供学习者参考。

图书在版编目(CIP)数据

C 语言程序设计教程/李明主编. —2 版. ——上海:
上海交通大学出版社,2012
21 世纪高等学校教材
ISBN 978-7-313-05004-5

I. C... II. 李... III. C 语言—程序设计—高
等学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字(2011)第 226907 号

C 语言程序设计教程

(第二版)

李 明 主编

上海交通大学出版社出版发行

(上海市番禺路 951 号 邮政编码 200030)

电话:64071208 出版人:韩建民

昆山市亭林印刷有限责任公司 印刷 全国新华书店经销

开本:787mm×1092mm 1/16 印张:20.5 字数:503 千字

2007 年 11 月第 1 版 2012 年 1 月第 2 版 2012 年 1 月第 4 次印刷

ISBN 978-7-313-05004-5/TP 定价:33.00 元

版权所有 侵权必究

告读者:如发现本书有印装质量问题请与印刷厂质量科联系
联系电话:0512-57751097

21 世纪高等学校教材

编审委员会

顾 问: 韩正之

执行主任: 百 文

副 主 任:	李 明	冷金麟	靳全勤	张华隆	蒋凤瑛
	冯 颖	普杰信	程全洲	潘群娜	杨裕根
	徐祖茂	张红梅	宓一鸣	姜献峰	李 敏
	李湘梅	闫洪亮	陈树平	包奇金宝	刘克成
	白丽媛	戴 兵	张占山	曹天守	

前 言

C 语言伴随着 Unix 的发展而诞生, 并以其概念简洁、数据类型丰富、结构清晰、表达能力强、功能强大、书写灵活等优势, 迅速在程序设计领域普及。它的模块化、结构化特性使其可以满足现代程序设计的需要, C 语言既可以用于书写系统软件, 也可用于书写应用软件, 还广泛应用于嵌入式系统开发。因此, C 语言成为当今软件开发领域中广泛应用的一种语言, 也是高等学校计算机语言类课程的首选。

本书自 2007 年出版以来, 被全国多所院校采用。结合具体的使用情况, 本次对教材进行了进一步的完善和修订。本书共分为 10 章, 系统全面地介绍了 C 语言的基本概念, C 语言的数据类型, C 语言的程序构成, 以及各种程序设计的方法。第 1 章介绍了 C 语言的基本概念和 C 程序的基本构成, 以及用 C 语言解决实际问题的基本步骤。第 2 章介绍了 C 语言的各种数据类型、运算符及其特性, 并阐述了数据的输入和输出函数。第 3 章介绍了程序的基本结构——顺序结构、分支结构、循环结构的基本设计思想及其使用。第 4 章对 C 语言的数组进行了系统的阐述。第 5 章介绍了通过指针访问数据的方法, 并重点描述了指针在数组中的应用。第 6 章介绍了用函数进行模块化程序设计的基本方法, 重点描述了模块的划分和函数之间数据的传递。第 7 章介绍了 C 语言提供的结构、联合、枚举和用户自定义类型, 以及利用它们进行程序设计的方法, 并描述了一种动态的数据结构——链表。第 8 章介绍了 C 语言提供的各种逻辑位运算和移位位运算。第 9 章介绍了 C 程序处理文件中的数据的基本方法。第 10 章以 C 语言的综合编程为实例, 详细阐述了用 C 语言开发应用软件的一般步骤和方法。

本书力图在注重培养学习者程序设计能力的同时, 提倡良好的程序设计的风格; 在重视语法结构介绍的同时, 使读者得到算法和方法上的提升。本书在介绍程序设计的基础上, 适当地介绍了软件工程的基本思想, 所有的案例力求做到符合现代程序设计的需要, 为学习者培养良好的程序设计的风格奠定了基础。本书各章均配有丰富的案例, 所有代码均在 Visual C++ 6.0 环境下调试通过, 可直接引用。

本书作者均有多年从事 C 语言以及相关课程教学的经验, 本书是作者在多次编写讲义和教材的基础上提炼而成的。第 1 章由李明执笔, 第 2、3 章由刘书影执笔, 第 4、8 章由余云执笔, 第 5、6 章由汪红霞执笔, 第 7、9 章由李春梅执笔, 第 10 章由胡贤德执笔。宣善立、李刚健老师也对本书的编写给予了全面的支持, 提供了大量的教学案例, 全书由李明负责统编定稿。

为配合本书的学习, 本书各章均配有一定数量的习题, 并配有 CAI 教学课件和配套的教材《C 语言程序设计实验指导与习题解答》, 供学习者参考。

由于作者才疏学浅, 书中的错误和不当之处, 恳请专家、读者批评指正。

编 者

2011 年 10 月

目 录

第1章 概述	1	2.4.4 关系运算符	33
1.1 C语言简介	1	2.4.5 逻辑运算符	34
1.2 C程序初探	3	2.4.6 条件运算符	35
1.2.1 简单的C程序	3	2.4.7 逗号运算符	36
1.2.2 C程序的基本结构	6	2.4.8 sizeof长度运算符	36
1.2.3 C程序的基本词汇	8	2.4.9 常用数学函数	36
1.3 算法及其描述	9	2.4.10 各种运算符的优先级	37
1.3.1 小试身手	9	2.4.11 数据类型的转换	37
1.3.2 算法的基本概念	13	2.5 基本的C语句	39
1.3.3 算法的各种描述方法	14	2.5.1 C语句	39
1.4 C程序的开发步骤	17	2.5.2 C语句分类	39
1.5 本章小结	18	2.6 数据的输入和输出	41
程序设计题1	18	2.7 字符数据的输入和输出	41
第2章 简单的C程序设计	19	2.7.1 putchar函数(字符输出函数)	41
2.1 C语言的数据及其类型	19	2.7.2 getchar函数(字符输入函数)	42
2.1.1 C语言的数据	19	2.7.3 变量获取数据的几种方法	42
2.1.2 C语言的数据类型	19	2.8 格式输入和输出	43
2.2 常量	20	2.8.1 printf函数	43
2.2.1 整型常量	20	2.8.2 scanf函数	46
2.2.2 实型常量	21	2.9 简单顺序程序设计举例	48
2.2.3 字符型常量	21	2.10 本章小结	51
2.2.4 字符串常量	22	程序设计题2	52
2.3 变量	22	第3章 分支和循环的C程序设计	53
2.3.1 变量及其说明	22	3.1 程序的基本结构	53
2.3.2 整型变量	24	3.2 选择结构	55
2.3.3 实型变量	26	3.2.1 if结构的3种形式	55
2.3.4 字符型变量	27	3.2.2 switch结构和break语句	61
2.4 常用运算符和表达式	28	3.2.3 各种分支的嵌套	64
2.4.1 运算符和表达式	28	3.3 循环结构	67
2.4.2 算术运算符	29	3.3.1 while结构	68
2.4.3 赋值运算符	32	3.3.2 do-while结构	70
		3.3.3 for结构	70

3.3.4 转移语句	73	程序设计题 4	122
3.3.5 各种循环的嵌套	74	第 5 章 指针	124
3.4 分支和循环的 C 程序设计	76	5.1 指针的基本概念	124
3.4.1 应用程序	76	5.1.1 地址和指针	124
3.4.2 两种程序设计的常用方法	85	5.1.2 指针的基本应用	125
3.5 结构化程序设计方法	86	5.1.3 指针的运算	127
3.5.1 关于 goto	86	5.2 指针与数值型数组	128
3.5.2 结构化程序设计的核心思想	87	5.2.1 指针与一维数组	128
3.5.3 自顶向下、逐步求精的程序设计方法	87	5.2.2 指针与多维数组	131
3.6 本章小结	90	5.3 指针与字符数组及字符串	135
程序设计题 3	90	5.3.1 字符串与指针	136
第 4 章 数组	91	5.3.2 字符数组与指针	137
4.1 数组的基本概念	91	5.4 指针数组和指向指针的指针	138
4.1.1 引例	91	5.4.1 指针数组	138
4.1.2 数组的基本概念	93	5.4.2 指向指针的指针	141
4.1.3 数组的分类	94	5.5 程序代码风格	143
4.2 一维数组	94	5.5.1 命名规范	143
4.2.1 一维数组的说明和引用	94	5.5.2 表达式及语句风格	144
4.2.2 一维数组的存储	96	5.6 本章小结	145
4.2.3 一维数组的初始化	96	程序设计题 5	146
4.2.4 一维数组的应用	98	第 6 章 函数	147
4.3 多维数组	99	6.1 函数的概念	147
4.3.1 二维数组的说明和引用	100	6.1.1 函数的基本概念	148
4.3.2 二维数组的存储	100	6.1.2 函数的分类及调用过程	149
4.3.3 二维数组的初始化	101	6.2 函数的定义与调用	150
4.3.4 二维数组的应用	101	6.2.1 函数的定义	150
4.4 字符数组	103	6.2.2 函数的调用	153
4.4.1 字符数组	103	6.2.3 函数调用数据的传递方式	156
4.4.2 字符数组的输入和输出	105	6.2.4 函数的原型声明	158
4.4.3 字符串处理函数	106	6.3 变量的作用域与生存期	159
4.5 数组应用举例	108	6.3.1 局部变量和全局变量	160
4.5.1 求极值问题	108	6.3.2 变量的存储类别	162
4.5.2 查找	111	6.3.3 关于变量存储类别的说明	167
4.5.3 排序	114	6.4 数组与函数参数	170
4.5.4 倒序	117	6.4.1 一维数组作为函数参数	170
4.6 算法与效率	119	6.4.2 二维数组作为函数参数	173
4.7 本章小结	122		

6.4.3 字符数组作为函数参数	175	7.8 枚举类型	227
6.5 返回指针的函数	176	7.8.1 枚举类型的定义	227
6.6 指向函数的指针	177	7.8.2 枚举变量的定义	227
6.7 递归函数	181	7.8.3 枚举变量的使用	228
6.8 主函数参数	185	7.9 用户定义类型	229
6.9 编译预处理	186	7.10 本章小结	229
6.9.1 宏定义	186	程序设计题 7	230
6.9.2 文件包含	191	第 8 章 位运算	231
6.9.3 条件编译	193	8.1 位运算基本概念	231
6.10 模块化程序设计方法	194	8.1.1 位运算的基本概念	231
6.11 本章小结	196	8.1.2 计算机中的数据表示	231
程序设计题 6	197	8.2 位运算符	233
第 7 章 结构和联合	198	8.2.1 逻辑位运算	234
7.1 引例	198	8.2.2 移位位运算	238
7.2 结构变量	200	8.3 位段	240
7.2.1 结构体类型的定义	200	8.4 位运算案例	241
7.2.2 结构体变量的定义和初始化	201	8.5 本章小结	243
7.2.3 结构体变量的使用	204	程序设计题 8	244
7.3 结构数组	206	第 9 章 文件	245
7.3.1 结构数组的定义和初始化	206	9.1 文件	245
7.3.2 结构数组的使用	207	9.1.1 引言	245
7.4 结构与指针	208	9.1.2 文件	246
7.4.1 指向结构体变量的指针	208	9.1.3 文件的分类	246
7.4.2 指向结构体数组的指针	210	9.1.4 文件指针	247
7.5 结构体与函数	211	9.1.5 文件的读写方式	248
7.5.1 结构体变量作为函数参数	211	9.2 文件的打开与关闭	248
7.5.2 指向结构体变量的指针作为 函数参数	212	9.2.1 文件的打开	248
7.5.3 返回结构体的函数	213	9.2.2 文件的关闭	250
7.6 动态内存与链表	215	9.3 文件的顺序读写	250
7.6.1 动态内存函数	215	9.3.1 字符输入输出函数	251
7.6.2 动态内存与链表的基本概念	217	9.3.2 字符串输入输出函数	255
7.6.3 链表的基本操作	218	9.3.3 格式输入输出函数	256
7.7 联合体	223	9.3.4 数据块输入输出函数	258
7.7.1 联合体类型的定义	223	9.3.5 整数输入输出函数	260
7.7.2 联合体变量的定义	224	9.3.6 标准设备文件的输入和输出	261
7.7.3 联合体变量的使用	225	9.4 文件的随机读写与定位	262

9.4.1	rewind 函数	262	10.2.1	问题的定义	278
9.4.2	ftell 函数	262	10.2.2	系统设计	279
9.4.3	fseek 函数	263	10.2.3	程序及运行结果	280
9.5	文件的错误检测	264	10.3	学生成绩管理系统	287
9.6	文件程序设计应用	266	10.3.1	需求分析	288
9.7	本章小结	268	10.3.2	概要设计	289
程序设计题 9	268		10.3.3	详细设计	290
第 10 章	综合实例	269	10.3.4	编码	292
10.1	几种数值计算方法	269	10.3.5	学生成绩管理系统源代码	294
10.1.1	定积分的数值计算	269	10.4	本章小结	304
10.1.2	一元方程根的数值计算	273	附录		305
10.1.3	蒙特卡洛法	276	参考文献		317
10.2	通过 C 语言程序实现复数的各类运算	278			

第 1 章 概 述

本章学习目标:

- 计算机、程序、语言
- C 语言的历史和特点
- C 语言程序的基本构成
- C 语言的算法及其描述
- C 语言的开发环境和程序开发的步骤

1.1 C 语言简介

1. 计算机与程序

从 20 世纪 40 年代电子计算机诞生以来, 计算机的应用领域几乎渗透到各行各业, 为人们做着各种各样的工作。为了让计算机完成各种工作, 人们必须首先编写程序。

程序是用计算机语言描述的、为解决某一问题、满足一定语法规则的语句序列。计算机是可以按照人们事先编写的程序高速、精确地进行数据加工、处理的电子装置。人们通过程序明确地告诉计算机做什么, 以及如何去做这些事情, 并将事先编写的程序存储到计算机内部, 启动计算机执行程序, 计算机在执行程序的过程中, 完成程序规定的任务。

2. 计算机的语言

为使计算机理解人类的意图, 必须书写程序, 而程序是用某种计算机语言书写的。计算机的语言有许多种, 可分为 3 大类: 机器语言、汇编语言和高级语言。

(1) 机器语言。机器语言是由 0 和 1 二进制代码按一定的规则组合而成的指令序列, 它是面向具体计算机系统的, 不同的计算机系统识别的机器语言是不同的。如: 在某种计算机系统中 00100101 代表加法; 10010101 代表减法。

机器语言能被计算机硬件直接理解和执行, 不需要另外的翻译软件, 占用空间少, 效率高。但其难于学习和记忆, 编程工作量大, 通用性差, 且随着不同的计算机系统而改变。

(2) 汇编语言。汇编语言是采用英文助记符表示的语言。如 add 代表加法, sub 代表减法。

与机器语言相比, 其学习和记忆难度有所下降, 但通用性仍然较差, 而且不同的计算机系统其汇编语言的指令也有所差别。用汇编语言编写的程序, 计算机硬件不能直接理解和执行, 需要另外的语言处理程序将其翻译成机器语言程序。

(3) 高级语言。高级语言是由表达各种意义的英文单词和数学式子按照一定的语法规则构成的语言。如: + 代表加法; - 代表减法。

高级语言接近于自然语言, 便于学习和记忆。用高级语言编写的程序, 计算机硬件也不能直接理解和执行, 需要另外的语言处理程序将其翻译成机器语言程序。

高级语言可分为面向过程的语言和面向对象的语言。面向过程的语言在程序中不仅要告诉计算机“做什么”，还要告诉计算机“怎么做”，即在程序中要详细描述用什么动作加工什么数据，即解题的过程和细节。用面向对象的语言编写程序时，在程序中只要告诉计算机“做什么”，一般不需要告诉计算机“怎么做”。

C 语言是面向过程的语言，C++语言是面向对象的语言。

3. C 语言的发展

C 语言的发展过程与 UNIX 操作系统是密不可分的。

1963 年英国的剑桥大学推出了 CPL(combined programming language)语言，并于 1967 年对其进行了简化，推出了 BCPL(basic combined programming language)语言；1970 年美国贝尔实验室的 Dennis Ritchie 和 Ken Thompson 以 BCPL 语言为基础，进行了进一步简化，推出了更简单的 B 语言，并用 B 语言书写了第一个操作系统 UNIX。在随后的几年中，贝尔实验室继续对 B 语言进行改进，并于 1973 年，推出了 C 语言，而且用 C 语言重新书写了 UNIX 操作系统，使其成为较完善的操作系统。因此人们称 Dennis Ritchie 和 Ken Thompson 为 C 语言的鼻祖。

以后人们对 C 语言进行了多次改进，使其功能日益完善，但一直在贝尔实验室内部使用。直到 1975 年，UNIX 推出了 6.0 版本后，C 语言的优点引起了人们的普遍关注，特别是 1977 年出现了不依赖具体机器的 C 语言编译文本《可移植 C 语言编译程序》，使得 C 语言移植到其他机器比较容易实现，这也使得 UNIX 操作系统迅速在各种机器上实现。1978 年出版的 C 语言的经典著作——《The C Programming Language》所给出的几乎是当时 C 语言的标准。

随着各种 C 语言版本的推出，为 C 语言制订标准成为必需。1983 年美国国家标准委员会对 C 语言进行了标准化，颁布了第一个 C 语言的标准草案——83 ANSI C；1987 年又颁布了新标准——87 ANSI C；1990 年，国际标准化组织 ISO(International Standard Organization)接受 87 ANSI C 为 ISO-C 的标准。目前流行的 C 编译系统都以此为基础。

80 年代后期，贝尔实验室又为 C 语言增加了面向对象的特性，即 C++。C 语言系统依然保留在 C++中，形成了 C++的子集。

4. C 语言的特点

尽管高级语言有很多，但由于 C 语言有着完善的功能，适用于多种编程要求，因而成为计算机专业人员首选的语言之一，也成为多数高校开设的程序设计课程首选的语言之一。其主要特点如下：

(1) 功能强大。C 语言功能强大，既可以用于书写系统软件，也可用于书写应用软件。

(2) 移植性强。在某种计算机系统上的 C 语言程序，只需作少量改动，甚至不作改动即可在其他的计算机系统中运行。

(3) 是主流的编程语言之一。C 语言是广大的计算机专业人员普遍采用的流行编程语言之一。

(4) 是模块化、结构化的程序设计语言。C 语言具有多种结构化的控制语句，可以实现各种功能。C 语言的源程序由一系列函数组成，它的模块化、结构化特性使其可以满足现代程序设计的需要。

(5) 有着丰富的运算符和数据类型。C 语言丰富的运算符和数据类型可以满足各种程序

设计的需要。

(6) 语法简洁、灵活，使用方便。

(7) 允许直接访问物理地址，具有一定的低级语言的特性。

(8) 程序生成的目标代码质量高，程序的执行效率高。

当然，C 语言也有其局限，如它的灵活性给编程人员带来自由的同时，可能也埋下了一定的风险；它的指针使得程序的执行过程难以跟踪；它的简洁使得程序难以阅读等。但鉴于其众多优点，C 语言仍然不失为人们首选的编程语言之一。

1.2 C 程序初探

1.2.1 简单的 C 程序

首先让我们暂时忽略细节来看几个简单的 C 语言程序的例子，并通过它们了解 C 语言的基本构成，从中了解和感受 C 语言程序设计的思想。

【例 1-1】字符串的输出。

编写程序代码 1-1 如下，其执行结果如图 1-1 所示。

代码 1-1

```
/*example 1-1 输出信息*/  
#include<stdio.h>  
void main(void)  
{  
    printf("欢迎走进 C 世界!");  
}
```

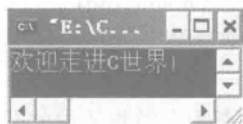


图 1-1 代码 1-1 的执行结果

程序分析：

(1) /*example 1-1 输出信息*/

此行为注释语句。在 C 程序中，/*注释内容*/语句为注释语句，注释内容不影响程序的执行，只是为了帮助读者阅读和理解程序。

(2) #include<stdio.h>

此行为文件包含，告诉系统必须包含文件 `stdio.h`，此文件为输入和输出提供支持。

(3) void main(void)

此句表明以下是一个 C 程序的主函数。

(4) {

此行表明主函数体由此开始。

(5) printf("欢迎走进 C 世界!");

此行为输出函数，其功能是在屏幕上显示双引号中的内容。

(6) }

此行表明主函数体到此结束。

程序功能：在屏幕上打印双引号中的信息“欢迎走进 C 世界!”。

【例 1-2】求 3 门课程的平均成绩。

编写程序代码 1-2 如下, 其执行结果如图 1-2 所示。

代码 1-2

```
/*example 1-2 计算平均数*/  
#include<stdio.h>  
void main(void)  
{  
    int mathscore,physscore,chemscore;           /*代表 3 门成绩*/  
    float avescore;                               /*代表平均成绩*/  
    mathscore = 75;  
    physscore = 85;  
    chemscore = 95;                               /*3 门成绩赋初值*/  
    avescore = (mathscore + physscore + chemscore) / 3.0; /*计算平均成绩*/  
    printf("avescore=%f",avescore);              /*输出平均成绩*/  
}
```

程序分析:

(1) /* example 1-2 计算平均数 */

此行为注释语句。

(2) #include<stdio.h>

此行为文件包含, 告诉系统必须包含文件 `stdio.h`。

(3) void main(void)

此行表明以下是一个 C 程序的主函数。

(4) {

此行表明主函数体由此开始。

(5) int mathscore, physscore, chemscore;

此行为说明语句, 说明本程序需要用这 3 个变量, 而且它们的数据类型是整型。

(6) float avescore;

此行为说明语句, 说明本程序需要用这个变量, 而且它的数据类型是浮点型。

(7) mathscore = 75;

hysscore = 85;

chemscore = 95;

此 3 行为赋值语句, 表明给这 3 个变量的初始值分别为: 75、85、95。

(8) avescore =(mathscore+physscore+chemscore)/3.0;

此行为赋值语句, 利用 3 个变量的初始值计算平均值。

(9) printf("avescore = %f", avescore);

此行为输出函数, 在屏幕上输出平均成绩。

(10) }

此行表明主函数体到此结束。

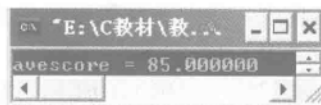


图 1-2 代码 1-2 的执行结果

程序功能: 程序中使用 4 个变量, 分别代表 3 门课程的成绩和平均成绩, 程序首先为 3 门课程的成绩赋值, 并将已知的 3 门课程的成绩求和后除以 3 来计算平均成绩, 利用 `printf` 函数将结果输出在屏幕上。

【例 1-3】求 2 个数中的大数，由主函数独立完成。
编写程序代码 1-3 如下，其执行结果如图 1-3 所示。

代码 1-3

```
/*example 1-3 求最大数*/
#include<stdio.h>
void main(void)
{
    int a,b,maxvalue;                /*定义 3 个变量*/
    printf("请输入 2 个数:");        /*输出提示信息*/
    scanf("%d%d",&a,&b);             /*由键盘输入 2 个数*/
    if(a > b)                         /*比较 2 个数的大小*/
        maxvalue = a;
    else
        maxvalue = b;
    printf("2 个数中的最大数是: %d",maxvalue); /*输出最大数*/
}
```

程序分析:

(1) int a, b, maxvalue;

此行为说明语句，说明本程序需要用这 3 个整型变量。

(2) printf("请输入 2 个数: ");

此行在屏幕上输出提示信息。

(3) scanf("%d%d", &a, &b);

此行利用输入函数，要求用户从键盘输入 2 个整数，分别存放到 *a* 和 *b* 的存储单元。

(4) if (a>b)

maxvalue = a;

else

maxvalue = b;

此 4 行为 if 语句，比较 2 个数的大小，取其中的大数给 maxvalue。

(5) printf("2 个数中的最大数是: %d ", maxvalue);

此行为输出函数，输出最大数。

程序功能：程序执行时，通过输入函数 scanf 临时由键盘输入 2 个数，赋给变量 *a* 和 *b*，接着对 2 个数进行大小的比较，根据比较的结果，将其中的大数放在变量 maxvalue 中，利用输出函数进行输出。

【例 1-4】求 2 个数中的大数，由主函数和子函数合作完成。

编写程序代码 1-4 如下，其执行结果如图 1-3 所示。

代码 1-4

```
/*example 1-4 利用子函数求最大数*/
#include<stdio.h>
int Getmax(int,int);                /*声明子函数*/
void main(void)
{
```

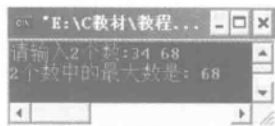


图 1-3 代码 1-3 的执行结果

```
int a,b,maxvalue;           /*定义 3 个变量*/
printf("请输入 2 个数:");    /*输出提示信息*/
scanf("%d%d",&a,&b);         /*由键盘输入 2 个数*/
maxvalue = Getmax(a,b);      /*调用子函数求最大数*/
printf("最大数是:%d",maxvalue); /*输出最大数*/
}

int Getmax(int x,int y)      /*定义子函数*/
{
    int z;
    if(x > y)                /*比较 2 个数的大小*/
        z = x;
    else
        z = y;
    return z;                /*返回最大数*/
}
```

程序分析:

(1) `int Getmax(int , int );`

声明子函数,表明以下有一个名为 `Getmax` 的子函数,它有 2 个整型参数,子函数的返回值为整型。

(2) `maxvalue = Getmax(a, b);`

此行为调用语句,调用名为 `Getmax` 的子函数,将参数 `a` 和 `b` 传递给子函数,并将返回结果赋给变量。

(3) `int Getmax(int x, int y)`

子函数定义,表明以下是名为 `Getmax` 的子函数。

(4) { (子函数中的大括号)

此行表明子函数体由此开始。

(5) `return z;`

返回语句,返回最大数。

(6) } (子函数中的大括号)

此句表明子函数体到此结束。

程序功能:程序将任务分为 2 部分:一部分负责数据的输入和输出,由称为 `main` 的主函数实现;另一部分负责在 2 个数中找出最大数,由称为 `Getmax` 的子函数实现。主函数采集数据,并通过参数将主函数中的变量 `a` 和 `b` 的值传递给子函数中的变量 `x` 和 `y`,子函数收到数据后,对它们进行判断,将结果放入变量 `z` 中,并将结果变量 `z` 返回给主函数,由主函数负责输出,即 2 个函数共同合作完成任务。

1.2.2 C 程序的基本结构

通过以上几个例子,虽然还不能完全掌握 C 程序,但可以基本了解 C 程序的基本构成。

1. 函数是构成 C 程序的基本单位

(1) C 程序是由函数组成, 其中有且仅有一个名为 `main` 的主函数, 根据实际编程的需要还可以有若干个子函数, 如 `Getmax` 为子函数, 函数是构成 C 程序的基本单位。

(2) 一个 C 程序总是从主函数 `main` 开始执行, 也是从主函数结束, 无论主函数书写在什么位置。

(3) 函数在执行过程中根据需要可以调用系统提供的库函数(如 `printf()`), 也可调用用户定义的函数(如 `Getmax()`)。如果调用系统提供的库函数, 在调用之前必须将相应的文件包含到本文件中; 如果调用用户定义的函数, 在调用之前必须声明。

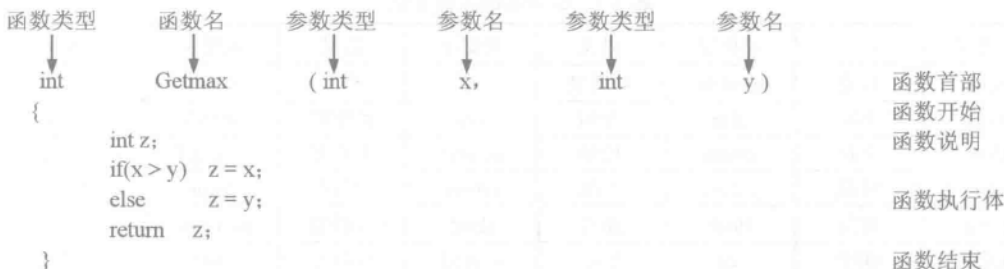
语句 `#include<stdio.h>` 和 `int Getmax(int, int);` 可解决该问题, 其详细内容将在后续章节中介绍。

2. 函数的组成

(1) 每一个函数由函数首部和函数体组成。

(2) 函数首部由函数类型、函数名、函数参数类型、函数参数名组成。

(3) 函数体由说明和执行 2 部分组成, 且函数体以左大括号开始, 以右大括号结束, 其构成如下:



3. 函数体的组成

(1) 函数体内由若干条 C 的语句组成。C 的语句有许多, 如: `int z;` 为说明语句; `c = a;` 为赋值语句; `return z;` 为返回语句; `/* ... */` 为注释语句等。

(2) C 的语句必须以分号 “;” 作为语句的结束。

(3) C 的语句书写比较自由, 如: 可在一行书写多条语句, 也可将一条语句写在多行, 但尽量一行只写一条语句, 以养成良好的、规范的程序设计风格。

(4) 与任何一种语言一样, C 的语句也必须满足一定的语法规则。

4. C 的语句的组成

C 程序的语句中含有各种符号、名称、数值等, 如: `int` 是英语单词 `integer`(整型)的缩写, 表示整型; `avescore =(mathscore+physscore+chemscore)/3.0;` 类似于数学式子; `x>y` 类似于不等式; `3.0` 代表一个数值; `return z` 代表的功能也与英文原意相同等等。

5. C 语言程序的书写字母

由于 C 语言区分大小写字母(一般使用小写字母), 因此命名时应特别注意。

1.2.3 C 程序的基本词汇

C 的语句中含有许多词汇和标识符号, 它们基本都来自于英语短语, 每一个标识符号都代表特定的含义, 它们是组成 C 语句的基本单位。

1. 字符集

字符是组成 C 语句的最小单位, C 程序中可以有以下几种字符:

(1) 英语字母: 大写字母 A~Z 和小写字母 a~z。

(2) 数字: 数字 0~9。

(3) 各种符号: 如 +、-、*、=、空格等。

这些符号构成了 C 语言基本的语法元素。

2. 关键字

关键字又称保留字, 是 C 语言预先规定的、具有固定意义的一些单词, 如: `int`、`return` 等, 用户只能按其预先规定的意义来使用它们, 不能改变其原来的意义。C 语言的关键字如表 1-1 所示。

表 1-1 C 语言的关键字表

关键字	语义	关键字	语义	关键字	语义	关键字	语义
<code>auto</code>	自动	<code>double</code>	双精度	<code>int</code>	整型	<code>struct</code>	结构
<code>break</code>	中断	<code>else</code>	否则	<code>long</code>	长整型	<code>switch</code>	开关
<code>char</code>	字符	<code>enum</code>	枚举	<code>register</code>	寄存器	<code>typedef</code>	自定义类型
<code>case</code>	情况	<code>extern</code>	外部	<code>return</code>	返回	<code>union</code>	共同体
<code>const</code>	常量	<code>float</code>	浮点	<code>short</code>	短整型	<code>unsigned</code>	无符号
<code>continue</code>	继续	<code>for</code>	循环	<code>signed</code>	有符号	<code>void</code>	空类型
<code>default</code>	缺省	<code>goto</code>	转向	<code>sizeof</code>	字节数	<code>volatile</code>	可变的
<code>do</code>	循环	<code>if</code>	如果	<code>static</code>	静态	<code>while</code>	循环

3. 标识符

C 语言的标识符有系统预定义标识符和用户自定义标识符, 一般用小写。

(1) 系统预定义标识符。系统预定义标识符是由系统预先定义的, 如: `main`、`printf` 等, 与关键字不同的是, 系统预定义标识符允许用户改变它的意义, 但这样做会失去其原有的含义, 从而造成误解, 因而这类标识符用户也不要更改其原意。

(2) 用户自定义标识符。用户自定义标识符是由用户自己根据需要定义的标识符, 通常用作变量名、数组名、函数名等, 如: `score`、`aver` 等, 标识符是满足以下规则的字符序列。

① 以字母或下划线开头。

② 由字母或下划线或数字组成。

③ 长度在一定范围的字符序列, 如: 一般环境只允许取 32 个字符, 建议不要太长。

④ 建议按照“见名知义”的原则, 如: `score` 代表分数, `mathscore` 代表数学分数, `aver` 代表平均数, `sum` 代表和等。