



普通高等教育“十二五”规划教材 计算机系列  
中国科学院教材建设专家委员会“十二五”规划教材

# C语言程序设计

C YUYAN CHENGXU SHEJI

金龙海 李 聪 主 编



科学出版社

普通高等教育“十二五”规划教材 计算机系列  
中国科学院教材建设专家委员会“十二五”规划教材

# C 语言程序设计

|     |     |         |
|-----|-----|---------|
| 金龙海 | 李 聪 | 主 编     |
| 王晓光 | 刘 威 | 李秀萍 副主编 |
| 苏莉蔚 | 姜 楠 | 参 编     |
| 丁继红 | 白洪涛 |         |

科 学 出 版 社

北 京

## 内 容 简 介

本书对 C 语言程序设计的基本知识、数据类型、结构化程序设计方法,以及常用算法及其应用进行了较为系统而详细的介绍。本书内容充实、体系完整、思路清晰、概念准确、选材新颖、注重实用,是编者总结多年教学经验编写而成的。

本书既可作为普通高等院校 C 语言程序设计课程教学用书,也可作为理工科各专业技术人员学习 C 语言程序设计的参考资料。

---

### 图书在版编目(CIP)数据

C 语言程序设计/金龙海,李聪主编. —北京:科学出版社,2012

ISBN 978-7-03-034324-6

I. ①C… II. ①金… ②李… 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(2012)第 096074 号

---

责任编辑:戴 薇 郭丽娜 / 责任校对:刘玉靖  
责任印制:吕春珉 / 封面设计:东方人华平面设计部

科 学 出 版 社 出版

北京东黄城根北街 16 号

邮政编码:100717

<http://www.sciencep.com>

北京九天志诚印刷有限公司 印刷

科学出版社发行 各地新华书店经销

\*

2012 年 8 月第 一 版 开本:787×1092 1/16

2012 年 8 月第一次印刷 印张:17 1/2

字数:418 000

定价:55.00 元(共两册)

(如有印装质量问题,我社负责调换<九天志诚>)

销售部电话 010-62142126 编辑部电话 010-62134021

版权所有,侵权必究

举报电话:010-64030229; 010-64034315; 13501151303

# 前 言

C 语言概念简洁,提供的数据类型和运算符较为丰富,表达能力强,使用灵活方便,目标代码执行效率高,程序结构性强,可读性和可移植性好,能充分体现结构化程序设计的风格和特点。C 语言既有高级语言的优点,又具有汇编语言的许多功能。因此,C 语言既适于应用软件的编写,又能用于系统软件的开发,是目前功能强大、应用范围广、影响力强的程序设计语言之一。

本书在全面而系统地介绍 C 语言的基本概念、语法功能、使用特性以及结构化程序设计方法和技能的基础上,通过具体实例分析,力求让学生理解和掌握数据结构、算法的概念及其应用;着重培养学生良好的程序设计思想和编程能力;为学生进一步学习 C 语言,更好地从事软件开发和工程应用打下坚实的基础。

本书的编者是多年从事 C 语言程序设计并在教学一线讲授计算机课程教学的老师。他们根据多年的教学心得和教改探索编写了本书,本书在编排体系、教学内容、教学目标、教学模式设计等方面,注重以现代教育理论为指导,以应用分析为基础,以多媒体教学为手段,力求体现“精讲多练”、“循序渐进”及程序设计语言教学的规律。书中所涉及的例题和习题也是经过精心挑选、编写和在 Visual C++ 6.0 环境下调试通过的,具有一定的代表性。

为了让学生更好地理解 C 语言的基本知识,提高编程能力,我们另编写了一本《C 语言程序设计实验指导与习题解答》(金龙海、刘威主编,科学出版社出版),可与本书配套使用。

本书共分 11 章,其中,金龙海编写第 5 章、第 8 章、第 11 章中的位运算、附录;王晓光编写第 1 章、第 7 章;刘威编写第 9 章、第 11 章中的预处理;李秀萍编写第 3 章、第 10 章;苏莉蔚编写第 6 章;姜楠编写第 4 章;丁继红编写第 2 章;白洪涛参与校稿。全书由金龙海负责统稿。

由于时间仓促,加之编者水平有限,书中难免存在错误和不当之处,恳请读者批评指正。

# 目 录

|                    |    |
|--------------------|----|
| 第 1 章 C 语言及算法概述    | 1  |
| 1.1 程序设计语言         | 1  |
| 1.1.1 计算机语言        | 1  |
| 1.1.2 计算机语言的发展     | 1  |
| 1.2 C 语言概述         | 3  |
| 1.2.1 C 语言的发展历史    | 3  |
| 1.2.2 C 语言的特点      | 4  |
| 1.3 C 语言程序初体验      | 5  |
| 1.4 算法概述           | 11 |
| 1.4.1 算法的概念        | 11 |
| 1.4.2 算法的特性        | 11 |
| 1.4.3 算法的表示方法      | 12 |
| 习题一                | 15 |
| 第 2 章 数据类型、运算符与表达式 | 18 |
| 2.1 C 语言的数据类型      | 18 |
| 2.2 常量             | 18 |
| 2.2.1 整型常量         | 19 |
| 2.2.2 实型常量         | 19 |
| 2.2.3 符号常量         | 19 |
| 2.2.4 字符型常量        | 20 |
| 2.2.5 字符串常量        | 21 |
| 2.3 变量             | 21 |
| 2.3.1 整型变量         | 23 |
| 2.3.2 实型变量         | 25 |
| 2.3.3 字符型变量        | 25 |
| 2.4 数值型数据间的混合运算    | 27 |
| 2.5 算术运算符和算术表达式    | 28 |
| 2.5.1 基本的算术运算符及表达式 | 28 |
| 2.5.2 自增运算符和自减运算符  | 29 |
| 2.6 赋值运算符和赋值表达式    | 30 |
| 2.6.1 赋值运算符        | 30 |
| 2.6.2 赋值类型转换       | 31 |
| 2.6.3 复合赋值运算符      | 33 |

|                                   |           |
|-----------------------------------|-----------|
| 2.6.4 赋值表达式 .....                 | 33        |
| 2.7 逗号运算符和逗号表达式 .....             | 34        |
| 习题二 .....                         | 35        |
| <b>第 3 章 顺序结构程序设计 .....</b>       | <b>38</b> |
| 3.1 C 语句概述 .....                  | 38        |
| 3.1.1 C 语句的作用与分类 .....            | 38        |
| 3.1.2 赋值语句 .....                  | 40        |
| 3.2 C 语言中数据的输入与输出 .....           | 41        |
| 3.2.1 数据输入/输出的概念及在 C 语言中的实现 ..... | 41        |
| 3.2.2 格式输出函数——printf .....        | 42        |
| 3.2.3 格式输入函数——scanf .....         | 49        |
| 3.2.4 字符输出函数——putchar .....       | 53        |
| 3.2.5 字符输入函数——getchar .....       | 54        |
| 3.3 顺序结构程序设计举例 .....              | 55        |
| 习题三 .....                         | 57        |
| <b>第 4 章 选择结构程序设计 .....</b>       | <b>61</b> |
| 4.1 关系运算 .....                    | 61        |
| 4.1.1 关系运算符 .....                 | 61        |
| 4.1.2 关系表达式 .....                 | 61        |
| 4.2 逻辑运算 .....                    | 62        |
| 4.2.1 逻辑运算符 .....                 | 62        |
| 4.2.2 逻辑表达式 .....                 | 62        |
| 4.3 if 语句 .....                   | 64        |
| 4.3.1 if 语句的一般形式 .....            | 64        |
| 4.3.2 if 语句的嵌套 .....              | 68        |
| 4.4 条件运算符与条件表达式 .....             | 69        |
| 4.5 switch 语句 .....               | 71        |
| 4.6 选择结构程序设计举例 .....              | 74        |
| 习题四 .....                         | 78        |
| <b>第 5 章 循环结构程序设计 .....</b>       | <b>81</b> |
| 5.1 while 语句 .....                | 81        |
| 5.2 do...while 语句 .....           | 82        |
| 5.3 for 语句 .....                  | 84        |
| 5.4 辅助控制语句 .....                  | 87        |
| 5.4.1 break 语句 .....              | 87        |
| 5.4.2 continue 语句 .....           | 88        |

|                   |            |
|-------------------|------------|
| 5.5 循环嵌套          | 89         |
| 5.6 goto 语句       | 90         |
| 5.7 循环结构程序设计举例    | 91         |
| 习题五               | 94         |
| <b>第 6 章 数组</b>   | <b>97</b>  |
| 6.1 一维数组          | 97         |
| 6.1.1 一维数组的定义和初始化 | 97         |
| 6.1.2 一维数组元素的引用   | 98         |
| 6.1.3 一维数组应用举例    | 99         |
| 6.2 二维数组          | 102        |
| 6.2.1 二维数组的定义     | 102        |
| 6.2.2 二维数组的初始化    | 103        |
| 6.2.3 二维数组元素的引用   | 104        |
| 6.2.4 二维数组应用举例    | 104        |
| 6.3 字符数组          | 107        |
| 6.3.1 字符数组的定义和初始化 | 107        |
| 6.3.2 字符数组的输入/输出  | 108        |
| 6.3.3 字符串处理函数     | 110        |
| 6.3.4 字符数组应用举例    | 113        |
| 习题六               | 115        |
| <b>第 7 章 函数</b>   | <b>118</b> |
| 7.1 函数的作用         | 118        |
| 7.2 C 程序的基本结构     | 118        |
| 7.3 函数分类          | 119        |
| 7.4 函数定义          | 120        |
| 7.4.1 函数定义的一般形式   | 120        |
| 7.4.2 函数的参数与返回值   | 121        |
| 7.5 函数的调用         | 123        |
| 7.5.1 函数的调用形式     | 123        |
| 7.5.2 对被调用函数的声明   | 125        |
| 7.6 函数的嵌套调用和递归调用  | 125        |
| 7.6.1 函数的嵌套调用     | 125        |
| 7.6.2 函数的递归调用     | 127        |
| 7.7 数组作为函数参数      | 128        |
| 7.7.1 数组元素作为函数参数  | 128        |
| 7.7.2 数组名作为函数参数   | 131        |
| 7.8 局部变量和全局变量     | 135        |

|                      |     |
|----------------------|-----|
| 7.8.1 局部变量           | 135 |
| 7.8.2 全局变量           | 137 |
| 7.9 变量的存储类别          | 139 |
| 7.9.1 静态存储方式和动态存储方式  | 139 |
| 7.9.2 局部变量的存储方式      | 139 |
| 7.9.3 全局变量的存储方式      | 141 |
| 7.9.4 存储类型总结         | 141 |
| 习题七                  | 142 |
| 第 8 章 指针             | 146 |
| 8.1 指针的基础知识          | 146 |
| 8.1.1 内存地址及指针        | 146 |
| 8.1.2 指针变量的定义        | 147 |
| 8.1.3 指针基本操作         | 148 |
| 8.1.4 指针变量作为函数参数     | 150 |
| 8.2 指针和数组            | 152 |
| 8.2.1 指针与一维数组        | 152 |
| 8.2.2 指针与二维数组        | 159 |
| 8.3 指针与字符串           | 163 |
| 8.3.1 用字符指针访问字符串     | 163 |
| 8.3.2 字符指针变量和字符数组的区别 | 165 |
| 8.3.3 字符指针变量作为函数参数   | 167 |
| 8.4 指针数组和指向指针的指针     | 169 |
| 8.4.1 指针数组           | 169 |
| 8.4.2 指向指针的指针        | 172 |
| 8.5 指针与函数            | 174 |
| 8.5.1 指针作为函数的返回值     | 174 |
| 8.5.2 指向函数的指针        | 174 |
| 习题八                  | 177 |
| 第 9 章 结构体与共用体        | 179 |
| 9.1 结构体概述            | 179 |
| 9.2 结构体类型变量          | 180 |
| 9.2.1 结构体变量的定义       | 180 |
| 9.2.2 结构体变量的引用及初始化   | 182 |
| 9.2.3 指向结构体变量的指针变量   | 185 |
| 9.3 结构体数组            | 187 |
| 9.3.1 结构体数组的定义及初始化   | 188 |
| 9.3.2 结构体数组元素的引用     | 189 |

|                       |     |
|-----------------------|-----|
| 9.3.3 指向结构体数组元素的指针变量  | 191 |
| 9.4 结构体类型数据作为函数参数     | 192 |
| 9.5 链表                | 193 |
| 9.5.1 链表概述            | 194 |
| 9.5.2 静态链表            | 195 |
| 9.5.3 动态链表            | 199 |
| 9.6 共用体               | 205 |
| 9.6.1 共用体类型定义         | 205 |
| 9.6.2 共用体变量的定义与引用     | 206 |
| 9.6.3 使用共用体类型数据时的注意事项 | 207 |
| 9.7 枚举类型              | 209 |
| 9.8 用 typedef 定义类型    | 210 |
| 习题九                   | 213 |
| 第 10 章 文件             | 219 |
| 10.1 C 语言文件概述         | 219 |
| 10.1.1 文件简介           | 219 |
| 10.1.2 文件的分类          | 220 |
| 10.1.3 缓冲文件系统         | 220 |
| 10.1.4 文件类型指针         | 221 |
| 10.2 文件操作函数           | 222 |
| 10.2.1 打开与关闭文件的函数     | 222 |
| 10.2.2 读/写数据文件内容的函数   | 225 |
| 10.2.3 文件的定位和随机读写     | 234 |
| 10.2.4 出错检测函数         | 236 |
| 习题十                   | 237 |
| 第 11 章 预处理和位运算        | 240 |
| 11.1 预处理              | 240 |
| 11.1.1 宏定义            | 240 |
| 11.1.2 文件包含           | 247 |
| 11.1.3 条件编译           | 250 |
| 11.2 位运算              | 252 |
| 11.2.1 位运算符           | 252 |
| 11.2.2 位运算应用举例        | 256 |
| 习题十一                  | 257 |
| 附录                    | 261 |
| 附录 1 常用字符与 ASCII 码对照表 | 261 |

|                        |     |
|------------------------|-----|
| 附录 2 C 语言的关键字及说明 ..... | 262 |
| 附录 3 运算符的优先级和结合性 ..... | 262 |
| 附录 4 常用的 C 语言库函数 ..... | 263 |
| 参考文献 .....             | 268 |

# 第 1 章 C 语言及算法概述

## 1.1 程序设计语言

### 1.1.1 计算机语言

人和人之间是通过语言进行交流的。美国人之间说英语，法国人之间说法语，中国人之间说汉语。同理，人和计算机交流信息也需要通过语言，这就需要创造一种计算机和人都能识别的语言，即计算机语言。

TIOBE 是一家评估和跟踪软件质量的专业公司，是著名的编程语言风向标。TIOBE 编程语言排行榜是一种编程语言的流行程度的指标，该指标每月更新一次，其市场占有率是根据世界各地的熟练工程师、课程和第三方供应商的数据统计的。根据 TIOBE 的数据，开发者可以看出自己的编程技能是否能跟上趋势，以及世界范围内开发语言的走势。表 1.1 是排在前 10 位的计算机编程语言。

表 1.1 TIOBE 编程语言排行榜前 10 位

| 2011 年 11 月 | 2010 年 11 月 | 编程语言           | 市场份额/% | 同比变更/% | 等级 |
|-------------|-------------|----------------|--------|--------|----|
| 1           | 1           | Java           | 17.87  | -0.63  | A  |
| 2           | 2           | C              | 17.32  | 0.61   | A  |
| 3           | 3           | C++            | 8.08   | -1.41  | A  |
| 4           | 5           | C#             | 7.32   | 1.61   | A  |
| 5           | 4           | PHP            | 6.10   | -1.72  | A  |
| 6           | 8           | Objective-C    | 5.98   | 2.79   | A  |
| 7           | 7           | (Visual) Basic | 5.04   | -0.43  | A  |
| 8           | 6           | Python         | 3.62   | -2.06  | A  |
| 9           | 10          | JavaScript     | 2.57   | 0.90   | A  |
| 10          | 9           | Perl           | 2.08   | -0.39  | A  |

从表 1.1 中可以看出，语言也如同其他生命体一样，有着自己的生命力，也同样面临着“物竞天择，适者生存”的境遇。C 语言从产生到现在，人们不仅在使用，而且是在广泛使用，一定有着其独特的“魅力”。并且，2011 年排在前四位的计算机编程语言，都是 C 语言的“近亲”，都与 C 语言有着千丝万缕的联系，也从另外一个角度说明了其影响力。

### 1.1.2 计算机语言的发展

计算机做的每一次动作、每一个步骤，都是按照已经用计算机语言编好的程序来执行的。程序是计算机要执行的指令的集合，而程序全部都是用我们所掌握的计算机语言来编写的。所以人们是通过计算机语言向计算机发出命令来控制计算机的。

计算机语言经历了以下几个发展阶段。

### 1. 机器语言

机器语言是直接由二进制代码指令表达的计算机语言。指令是由 0 和 1 组成的一串代码，有一定的位数，并分成若干段。各段的编码表示不同的含义，如某台计算机字长为 16 位，即由 16 个二进制数组成一条指令或其他信息。16 个 0 和 1 可组成各种排列组合，通过线路变成电信号，让计算机执行各种不同的操作。

限于目前的科技水平，计算机全称还叫“电子计算机”。而电子元器件有两种稳定的工作状态，那就是“开”和“关”。所以从物理上决定了目前电子计算机采用二进制进行运算，计算机唯一所能识别的语言也只有机器语言，即由 0 和 1 构成的代码。但由于其难以记忆和识别，出错的时候也难以修改，所以通常人们不采用机器语言编程。

### 2. 汇编语言

汇编语言是面向机器的程序设计语言。在汇编语言中，用助记符代替操作码，用地址符号（或标号）代替地址码，通过用符号代替机器语言中的二进制码，可以把机器语言变成汇编语言，汇编语言也称为符号语言。汇编语言的实质和机器语言是相同的，都是直接对硬件操作，只不过指令采用了英文缩写的标识符，更容易识别和记忆，即汇编语言实际上就是机器语言的符号化。汇编程序的每一句指令都对对应实际操作过程中的一个动作，如“ADD”代表加，“MOV”代表数据传送等，这样人们很容易读懂并理解程序在干什么，纠错及维护都变得方便了，但它仍然需要编程者将每一步具体的操作命令的形式写出来，而且需要编程者对计算机的硬件有充分的了解，因此汇编源程序一般比较冗长、复杂、容易出错。但正因为汇编语言直接对硬件进行操作，所以其优点也是显而易见的，那就是汇编语言源程序经汇编生成的可执行文件不仅比较小，而且执行速度很快。反过来，汇编语言也同样由于与硬件紧密相关，所以其移植性比较差。

机器语言和汇编语言都是面向机器的语言，由于它们“贴近”计算机，所以称为低级语言。

### 3. 高级语言

为了克服低级语言的缺点，20 世纪 50 年代中期出现了高级语言。高级语言主要是相对于机器语言和汇编语言而言的，它并不是特指某一种具体的语言，而是包括了很多编程语言，如 FORTRAN、BASIC、COBOL、Pascal、C 等。这些语言的语法、命令格式都各不相同。

高级语言是目前绝大多数编程者的选择。和汇编语言相比，它不仅将许多相关的机器指令合成为单条指令，并且去掉了与具体操作有关但与完成工作无关的细节。例如，使用堆栈、寄存器等，大大简化了程序中的指令。同时，由于省略了很多细节，编程者也就不需要有太多的计算机硬件知识。

高级语言所编制的程序不能直接被计算机识别，必须经过转换才能被执行，按转换方式可将它们分为两类。

1) 解释类：执行方式类似于日常生活中的“同声翻译”，应用程序源代码一边由相应语

言的解释器“翻译”成目标代码（机器语言），一边执行，因此效率比较低，而且不能生成可独立执行的可执行文件，应用程序不能脱离其解释器。但这种方式比较灵活，可以动态地调整、修改应用程序，如 BASIC 语言程序。

2) 编译类：执行方式是将程序源代码经编译程序“翻译”成目标代码（二进制），再经过连接程序连接，变成可执行文件运行。可执行文件是可以脱离其语言环境独立执行的，因此使用比较方便、效率较高。但应用程序一旦需要修改，必须先修改源代码，再重新通过编译、连接变成可执行文件才能执行。大多数的高级语言程序都是编译型的，如 FORTRAN、C、Pascal 等。

此时的程序设计语言都属于过程化语言，是面向过程的，在编写程序时需要具体指定每一个过程的细节。在编写规模较小的程序时，使用这些编程语言还能得心应手，但在处理规模较大的程序时，就显得捉襟见肘、力不从心了。

#### 4. 基于面向对象思想的非过程化语言

为了解决面向过程的程序设计的缺点，20 世纪 80 年代初期开始出现了面向对象程序设计概念。面向对象程序设计在程序设计模式中是一个新的概念，从现实世界中客观存在的事物（对象）出发来构造软件系统，并在系统构造中尽可能运用人类的自然思维方式，强调直接以问题域（现实世界）中的事物为中心来思考问题，认识问题，并根据这些事物的本质特点，把它们抽象地表示为系统中的对象，作为系统的基本构成单位。一个类的定义中不仅仅包括了此类对象应该共有的静态属性（类似于后面将要讲到的 C 语言中的结构体），而且还包括其动态方法。例如，定义了一个汽车类，该类中不仅包括车身形状、颜色、尺寸、重量，而且还包括发动、行驶、刹车、拐弯等动作。面向对象编程语言是基于事件驱动模型和消息传递方式来组织程序的。

从编程方式上看，面向对象编程语言主要特点是采用图形窗口和人机对话形式，基于数据库和“面向对象”技术，易编程、易使用、易维护。例如，常用的 Visual Basic、Java 等。再如美国国家仪器公司（national instruments, NI）的 LabView 则完全采用基于流程图的图形化编程方式，因此也被称为 G 语言（graphical language）。

## 1.2 C 语言概述

### 1.2.1 C 语言的发展历史

#### 1. C 语言的诞生

C 语言的发展颇为有趣，它的原型是 ALGOL 60 语言（也称为 A 语言）。1963 年，剑桥大学将 ALGOL 60 语言发展成为 CPL（combined programming language）语言。1967 年，剑桥大学的 Martin Richards 对 CPL 语言进行了简化，于是产生了 BCPL 语言。1970 年，美国贝尔实验室的 Ken Thompson 将 BCPL 进行了修改，并为它起了一个有趣的名字“B 语言”，

意思是将 CPL 语言“煮干”，提炼出它的精华，并且他用 B 语言编写了第一个 UNIX 操作系统。而在 1973 年，B 语言也被人“煮”了一下，美国贝尔实验室的 D.M.RITCHIE 在 B 语言的基础上最终设计出了一种新的语言，他取了 BCPL 的第二个字母作为这种语言的名字，这就是 C 语言。1978 年，Brian W.Kernighan 和 Dennis M.Ritchie 出版了名著 *The C Programming Language*，从而使 C 语言成为目前世界上流行的高级程序设计语言。

## 2. C 语言的发展

1987 年，随着微型计算机的日益普及，出现了许多 C 语言版本。由于没有统一的标准，这些 C 语言之间出现了一些不一致的地方，为了改变这种情况，美国国家标准化协会(ANSI)为 C 语言制定了一套 ANSI 标准，成为现行的 C 语言标准。1990 年，国际标准化组织(ISO)接受了 87 ANSI C 为 ISO C 的标准，1994 年，ISO 修订了 C 语言的标准。目前流行的 C 语言编译系统大多是以 ANSI C 为基础进行开发的，但不同版本的 C 语言编译系统所实现的语言功能和语法规则会稍有差别。C 语言发展迅速，而且成为很受欢迎的语言之一，主要因为它具有强大的功能。许多著名的系统软件，如 dBASE III PLUS、dBASE IV 都是由 C 语言编写的，用 C 语言加上一些汇编语言子程序，就更能显示 C 语言的优势了，像 PC-DOS、WORDSTAR 等就是用这种方法编写的。

### 1.2.2 C 语言的特点

个性是人的存在方式，正如尼采所说“世界上本来就没有相同的东西”。C 语言作为一种计算机语言，有其个性、独特之处，也有其不足的地方。下面对 C 语言的特点做一个简单的介绍。

#### 1. 简洁紧凑、灵活方便

C 语言一共只有 32 个关键字，9 种控制语句，程序书写自由，主要用小写字母表示。C 语言以接近英语国家的自然语言和数学语言为语言的表达形式，容易理解。C 语言把高级语言的基本结构和语句与低级语言的实用性结合起来，可以像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元。

#### 2. 运算符丰富

C 语言的运算符包含的范围很广泛，共有 34 个运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理，从而使 C 语言的运算类型极其丰富，表达式类型多样化。灵活使用各种运算符可以实现在其他高级语言中难以实现的运算。

#### 3. 数据类型丰富

C 语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等，能用来实现各种复杂数据类型的运算。C 语言也引入了指针概念，使程序效率更高。

#### 4. 结构化程序设计语言

C语言具有结构化程序语言所要求的三大基本结构，层次清晰，逻辑性强，便于维护、调试。

#### 5. 语法限制不太严格，程序设计自由度大。

一般的高级语言语法检查比较严，能够检查出几乎所有的语法错误，而C语言允许程序编写者有较大的自由度，客观上降低了对程序员的要求，但这种不严格事实上也给程序留下了出现一些潜在错误的可能性，降低了程序的健壮性。

#### 6. 允许直接访问物理地址，直接对硬件进行操作

在计算机世界中，位（bit）是最小单位，1 bit 就是1个二进制位，只有两种取值，即0和1。C语言能进行位运算，能实现汇编语言的大部分功能，能对硬件直接操作。很多嵌入式系统中的单片机都提供C语言编译器，如51系列单片机、MSP430、ARM等。

#### 7. 程序生成代码质量高，程序执行效率高

机器语言是能被计算机直接执行的语言，效率最高。汇编语言次之，基本接近于机器语言的效率。而C语言一般只比汇编程序生成的目标代码效率低10%~20%。

#### 8. C语言适用范围大，可移植性好

C语言的一个突出的优点就是适合于多种操作系统，如DOS、UNIX，也适用于多种机型。用C语言编写的程序不需要做很多改动就可以从一种机型上移到另一种机型上运行。

总之，C语言既具有高级语言的特点，又具有低级语言的特点；既是一个成功的系统设计语言，又是一个实用的程序设计语言；既能用来编写不依赖计算机硬件的应用程序，又能用来编写各种系统程序。尽管C语言也有不足，如对数组下标越界不做检查、对变量类型约束不严格等，但仍是一种很受欢迎、应用广泛的程序设计语言。

### 1.3 C语言程序初体验

计算机只能识别由0和1组成的机器语言。C语言源程序想要被计算机执行要经过编辑、编译、连接、运行四个步骤。这四个步骤完全可以使用不同的工具来完成，即可以用任何一个文本编辑软件来编辑源文件，用编译命令来编译，用连接命令来连接，如果没有错误，就可以运行了。但这样做太过烦琐，所以很多软件把这几个功能集成到了一起，形成集成开发环境（IDE）。

C语言的集成开发环境有很多种，典型的有TC 3.0、Borland C++、C++ Builder、Dev C++、Visual C++等。下面重点介绍Visual C++ 6.0（以下简称VC），VC是Microsoft公司的Visual Studio开发工具箱中的一个C++程序开发包，从最早期的1.0版本，发展到最新的6.0版本，VC已经有了很大的变化，在界面、功能、库支持方面都有许多的增强。最新的6.0版本在

编译器、MFC 类库、编辑器以及联机帮助系统等方面都比以前的版本做了较大改进。

这里需要说明两点：①VC 本来是作为 C++ 的集成调试环境，向下兼容也可以调试 C 语言程序，初学者只需要用到其中非常少的一部分功能；②之所以选择 VC，主要是考虑其应用的广泛性，另外国家计算机等级考试 C 语言二级和三级都采用此编程环境。

图 1.1 是 VC 的启动界面，下面简单介绍如何用 VC 来生成第一个 C 语言程序。

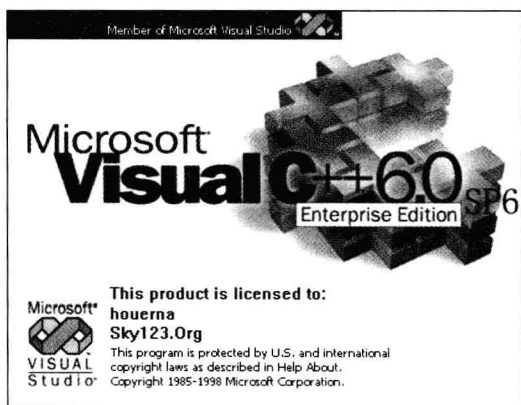


图 1.1 VC 的启动界面

运行一个 C 语言程序的一般过程如下：

1) 启动 VC，进入 VC 集成开发环境。

2) 编辑（或修改）源程序。

3) 编译。如果编译成功，则可进行下一步操作；否则，返回步骤 2) 修改源程序，再重新编译，直至编译成功。

4) 连接。如果连接成功，则可进行下一步操作；否则，根据系统的错误提示，进行相应修改，再重新连接，直至连接成功。

5) 运行。通过观察程序运行结果，验证程序的正确性，如果出现逻辑错误，则必须返回步骤 2) 修改源程序，再重新编译、连接和运行，直至程序正确。

6) 退出 VC 集成开发环境，结束本次程序运行。

下面就以实例来完整描述运行 C 语言程序的过程。

### 1. 启动 VC，进入源程序编辑界面

#### (1) 启动 VC

VC 是 Visual Studio 开发平台的一个组成部分，读者可以在 Windows 的程序项中找到 VC 的启动图标，通过它或者桌面的快捷方式，可以启动 VC 的编辑环境。

#### (2) 建立 C 语言程序文件

1) 单击“File”（文件）菜单中的“New”（新建）选项。

2) 在弹出的“New”对话框中选择“Files”选项卡，然后选择“C++ Source File”选项。

3) 输入相应的文件名和文件存放目录，如图 1.2 所示，并单击“OK”按钮。

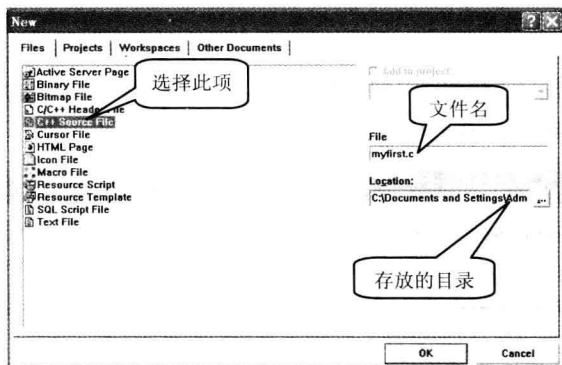


图 1.2 “New”对话框

## 2. 编辑 C 语言程序

进入 VC 编辑窗口后，输入例 1.1 所示的程序，如图 1.3 所示。



图 1.3 录入例 1.1 所示程序

**【例 1.1】** 输出一行字符串“这是我的第一个 C 语言程序”后换行。根据题意，编写如下的 C 语言程序。

```
/* #####  
   C 程序初体验  
   ##### */  
#include<stdio.h>    //包含标准输入/输出头文件  
void main( )  
{  
    printf("这是我的第一个 C 语言程序\n");  
}
```

在进行下一步之前，请注意窗口上的一个细节，如果一个程序编辑后还没有保存，文件名后面就会有一个“\*”号，单击“保存”或者“编译”按钮后，“\*”号就会消失。