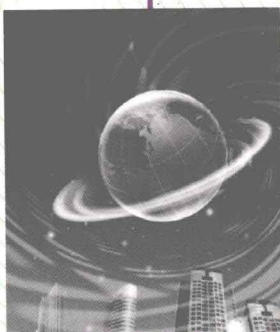
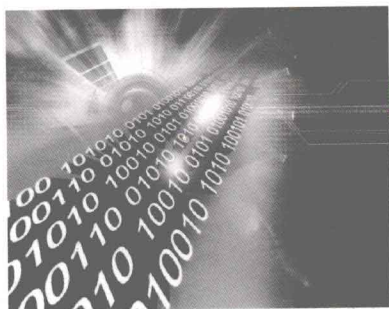


高等学校规划教材·计算机实用软件应用系列教程

# C 语言 程序设计实用教程

主编 曹岩 刘佳



西北工业大学出版社

高等学校规划教材·计算机实用软件应用系列教程

# C 语言程序设计实用教程

主编 曹 岩 刘 佳

西北工业大学出版社

**【内容简介】**本书系统全面地介绍了 C 语言程序设计的基本概念、方法和过程。主要内容包括：绪论，Turbo C 2.0 集成开发环境，数据类型、运算符和表达式，顺序结构，选择结构，循环结构，数组，函数，编译预处理，指针，结构与链表，联合、枚举、堆栈、队列与二叉树，文件，图形与声音，Visual C++ 6.0 集成开发环境，C 语言程序开发综合实例，附录等。

本书内容全面实用，实例丰富，适合各类高校学生作为教材或参考书使用，也可供科学与工程计算、数值计算、图形处理、自动控制、科学绘图、数据分析、设计、制造、控制、机械、电子、电器等领域从事 C 语言程序设计的科学研究和工程技术人员参阅。

#### 图书在版编目（CIP）数据

C 语言程序设计实用教程/曹岩，刘佳主编. —西安：西北工业大学出版社，2011.1  
高等学校规划教材·计算机实用软件应用系列教程  
ISBN 978-7-5612-2996-5

I. ①C… II. ①曹…②刘… III. ①C 语言—程序设计—教材 IV. ①TP312

中国版本图书馆 CIP 数据核字（2011）第 009734 号

出版发行：西北工业大学出版社

通信地址：西安市友谊西路 127 号 邮编：710072

电 话：（029）88493844 88491757

网 址：www.nwpup.com

电子邮箱：computer@nwpup.com

印 刷 者：陕西兴平报社印刷厂

开 本：787 mm×1 092 mm 1/16

印 张：24.25

字 数：652 千字

版 次：2011 年 1 月第 1 版

2011 年 1 月第 1 次印刷

定 价：40.00 元

# 前 言

C 语言作为一种结构化的计算机程序设计语言,是目前最流行和使用最广泛的计算机语言之一,其以强大的功能在系统软件开发、科学计算、自动控制等各个领域有着十分广泛的应用。C 语言具有表达能力强,概念和功能丰富,目标代码质量高,使用灵活方便,可移植性好等特点。C 语言既具有高级语言的优点,又具有低级语言的特点,特别适合于编写系统软件和嵌入式软件。

C 语言是面向过程的高级程序设计语言,而高级程序设计语言是当今大学生的必修课程,在绝大多数高等院校几乎所有的专业都把 C 语言作为首选程序设计语言课程。全国计算机等级考试、全国计算机应用技术证书考试也都将 C 语言列入了考试范围。尽管 C 语言具有其他高级程序设计语言不可比拟的优点,但是 C 语言的概念多、规则复杂、知识丰富、学习量大,因此容易出错。学生往往感觉掌握困难,普遍反映难学、难懂,理论与实践脱节。

本书从使用者的角度出发,系统介绍了 C 语言程序设计的基本概念、方法和过程。全书共包括 17 章。第 1 章绪论,第 2 章 Turbo C 2.0 集成开发环境,第 3 章数据类型、运算符和表达式,第 4 章顺序结构,第 5 章选择结构,第 6 章循环结构,第 7 章数组,第 8 章函数,第 9 章编译预处理,第 10 章指针,第 11 章结构与链表,第 12 章联合、枚举、堆栈、队列与二叉树,第 13 章文件,第 14 章图形与声音,第 15 章 Visual C++ 6.0 集成开发环境,第 16 章 C 语言程序开发综合实例(1),第 17 章 C 语言程序开发综合实例(2),附录等。

本书内容全面实用,实例丰富,采用循序渐进、通俗易懂的讲解方法以及理论与实际相结合的原则,在阐明 C 语言的基本概念、语法与正确使用的基础上,通过实例系统地讲述 C 语言程序设计的基本方法、规则和技巧,使学生掌握利用 C 语言进行结构化程序设计的技术和方法。每一章首先给出 C 语言程序设计的学习内容,然后通过一系列实例进行详细阐述,以加深对知识点的理解,提高分析问题和设计算法的能力。

全书由曹岩,刘佳主编。其中,第 1, 3, 4, 7 章以及附录由西安科技大学刘佳编写,第 2 章由河北工程大学魏红君编写,第 5, 6 章由西安科技大学付立东编写,第 8, 9 章由西安科技大学高峰编写,第 10 章由西安科技大学甘凯编写,第 11, 12 章由西安科技大学贾建华编写,第 13 章由西安科技大学靳红梅编写,第 14 章由西安科技大学梁荣编写,第 15 章由河北政法职业学院祝振欣编写,第 16, 17 章由西安科技大学李海文编写。参编人员还有丁雪芳、白瑀、杜江、曹森等。全书由曹岩,刘佳统稿。

本书适合各类高校学生作为教材或参考书使用,也可供科学与工程计算、数值计算、图形处理、自动控制、科学绘图、数据分析、设计、制造、控制、机械、电子、电器等领域从事 C 语言程序设计的科学研究和工程技术人员参阅。

由于水平及使用经验有限,疏漏之处在所难免,望各位读者不吝赐教,笔者在此深表感谢。

作者

2009 年 10 月

# 目 录

|                             |    |                         |    |
|-----------------------------|----|-------------------------|----|
| 第1章 绪论.....                 | 1  | 2.3.7 帮助.....           | 30 |
| 1.1 C语言的发展.....             | 1  | 2.3.8 项目管理.....         | 30 |
| 1.2 C语言版本.....              | 2  | 2.4 程序的编辑及调试.....       | 32 |
| 1.3 C语言的特点.....             | 2  | 2.5 常见编程错误.....         | 34 |
| 1.4 C语言的基本词法.....           | 3  | 2.6 应用实例.....           | 38 |
| 1.4.1 字符集.....              | 3  | 思考题.....                | 41 |
| 1.4.2 关键字.....              | 3  | 第3章 数据类型、运算符和表达式.....   | 42 |
| 1.4.3 标识符.....              | 3  | 3.1 C语言的数据类型.....       | 42 |
| 1.5 C程序的基本结构和结构特点.....      | 4  | 3.1.1 C语言的数据类型简介.....   | 42 |
| 1.5.1 C程序的基本结构.....         | 4  | 3.1.2 常量和变量.....        | 43 |
| 1.5.2 C语言的结构特点.....         | 6  | 3.1.3 整型数据.....         | 44 |
| 1.6 C语言程序开发过程.....          | 7  | 3.1.4 实型数据.....         | 48 |
| 1.7 C语言程序开发环境.....          | 8  | 3.1.5 字符型数据.....        | 49 |
| 1.8 算法.....                 | 9  | 3.1.6 变量的初始化.....       | 53 |
| 1.8.1 算法的概念和特性.....         | 9  | 3.1.7 数据类型之间的转换与运算..... | 53 |
| 1.8.2 算法的表示方法.....          | 10 | 3.2 算术运算符和算术表达式.....    | 56 |
| 思考题.....                    | 18 | 3.2.1 C语言的运算符简介.....    | 56 |
| 第2章 Turbo C 2.0 集成开发环境..... | 19 | 3.2.2 算术运算符和算术表达式.....  | 56 |
| 2.1 Turbo C 2.0 简介.....     | 19 | 3.3 赋值运算符和赋值表达式.....    | 58 |
| 2.2 Turbo C 2.0 的安装与设置..... | 19 | 3.3.1 赋值运算符类型.....      | 58 |
| 2.2.1 系统配置要求.....           | 19 | 3.3.2 赋值运算符和表达式.....    | 59 |
| 2.2.2 TC 2.0 的安装.....       | 19 | 3.4 其他运算符和表达式.....      | 60 |
| 2.2.3 TC2.0 的启动.....        | 20 | 3.4.1 逗号运算符和表达式.....    | 60 |
| 2.2.4 工作目录的设置.....          | 21 | 3.4.2 关系运算符和表达式.....    | 60 |
| 2.2.5 环境设置.....             | 22 | 3.4.3 逻辑运算符和表达式.....    | 60 |
| 2.3 系统功能.....               | 23 | 3.4.4 条件运算符和表达式.....    | 61 |
| 2.3.1 工作界面.....             | 23 | 3.5 实例.....             | 61 |
| 2.3.2 菜单.....               | 24 | 思考题.....                | 63 |
| 2.3.2 配置文件.....             | 28 | 第4章 顺序结构.....           | 64 |
| 2.3.3 编译器.....              | 28 | 4.1 顺序结构程序设计.....       | 64 |
| 2.3.4 链接.....               | 28 | 4.2 C语句概述.....          | 64 |
| 2.3.5 预处理.....              | 29 | 4.3 基本输入/输出处理.....      | 66 |
| 2.3.6 库管理.....              | 29 | 4.3.1 字符输入/输出函数.....    | 67 |

|                                 |            |                          |            |
|---------------------------------|------------|--------------------------|------------|
| 4.3.2 格式化输入/输出函数 .....          | 68         | 7.4.3 字符数组的引用 .....      | 111        |
| 4.4 实例 .....                    | 72         | 7.4.4 字符数组的输入/输出 .....   | 111        |
| 思考题 .....                       | 73         | 7.4.5 字符串处理函数 .....      | 113        |
| <b>第 5 章 选择结构</b> .....         | <b>74</b>  | 7.5 实例 .....             | 116        |
| 5.1 if 选择结构 .....               | 74         | 思考题 .....                | 118        |
| 5.1.1 单分支的 if 语句 .....          | 74         | <b>第 8 章 函数</b> .....    | <b>120</b> |
| 5.1.2 双分支的 if 语句 .....          | 76         | 8.1 概述 .....             | 120        |
| 5.1.3 多分支选择语句 .....             | 77         | 8.2 函数的定义 .....          | 121        |
| 5.2 switch 选择结构 .....           | 80         | 8.2.1 无参函数定义的一般形式 .....  | 121        |
| 5.3 break 语句 .....              | 81         | 8.2.2 有参函数定义的一般形式 .....  | 122        |
| 5.4 选择结构嵌套 .....                | 83         | 8.3 函数的参数和函数的返回值 .....   | 123        |
| 5.5 实例 .....                    | 86         | 8.3.1 函数的参数 .....        | 123        |
| 思考题 .....                       | 90         | 8.3.2 函数的返回值 .....       | 124        |
| <b>第 6 章 循环结构</b> .....         | <b>91</b>  | 8.4 函数的调用 .....          | 125        |
| 6.1 for 循环结构 .....              | 92         | 8.4.1 函数调用的方法 .....      | 125        |
| 6.2 while 语句和 do-while 语句 ..... | 94         | 8.4.2 函数调用时参数间的传递 .....  | 126        |
| 6.2.1 while 语句 .....            | 94         | 8.4.3 函数声明的方法 .....      | 131        |
| 6.2.2 do-while 语句 .....         | 96         | 8.5 函数的嵌套与递归调用 .....     | 132        |
| 6.3 continue 语句和 break 语句 ..... | 97         | 8.5.1 函数的嵌套调用 .....      | 132        |
| 6.3.1 continue 语句 .....         | 97         | 8.5.2 函数的递归调用 .....      | 133        |
| 6.3.2 break 语句 .....            | 97         | 8.6 变量的作用域 .....         | 135        |
| 6.4 goto 语句 .....               | 99         | 8.6.1 局部变量 .....         | 135        |
| 6.5 循环的嵌套 .....                 | 100        | 8.6.2 全局变量 .....         | 136        |
| 6.6 实例 .....                    | 102        | 8.7 变量的存储类别 .....        | 138        |
| 思考题 .....                       | 105        | 8.7.1 自动变量 .....         | 139        |
| <b>第 7 章 数组</b> .....           | <b>106</b> | 8.7.2 外部变量 .....         | 141        |
| 7.1 数组的基本概念 .....               | 106        | 8.7.3 静态变量 .....         | 141        |
| 7.2 一维数组 .....                  | 106        | 8.7.4 寄存器变量 .....        | 143        |
| 7.2.1 一维数组的定义和初始化 .....         | 106        | 8.8 内部函数和外部函数 .....      | 144        |
| 7.2.2 一维数组的引用 .....             | 107        | 8.9 要点 .....             | 145        |
| 7.3 二维数组及多维数组 .....             | 108        | 思考题 .....                | 145        |
| 7.3.1 二维数组的定义和初始化 .....         | 108        | <b>第 9 章 编译预处理</b> ..... | <b>147</b> |
| 7.3.2 二维数组的引用 .....             | 109        | 9.1 宏定义 .....            | 147        |
| 7.3.3 多维数组 .....                | 110        | 9.1.1 无参宏定义 .....        | 147        |
| 7.4 字符数组 .....                  | 110        | 9.1.2 带参宏定义 .....        | 150        |
| 7.4.1 字符数组的定义 .....             | 110        | 9.2 文件包含 .....           | 153        |
| 7.4.2 字符数组的初始化 .....            | 110        | 9.3 条件编译 .....           | 155        |

|                                |     |                                          |     |
|--------------------------------|-----|------------------------------------------|-----|
| 思考题 .....                      | 158 | 11.8 用指针和结构构成链表 .....                    | 197 |
| <b>第 10 章 指针</b> .....         | 159 | 11.8.1 链表的基本概念 .....                     | 197 |
| 10.1 指针的基本概念 .....             | 159 | 11.8.2 链表的操作 .....                       | 201 |
| 10.2 指针变量的类型说明 .....           | 160 | 11.8.3 单向链表的建立、输出、删除<br>与插入 .....        | 201 |
| 10.3 指针变量的赋值 .....             | 160 | 11.8.4 双向链表的建立、输出、删除<br>与插入 .....        | 208 |
| 10.4 指针变量的运算 .....             | 161 | 11.9 实例 .....                            | 213 |
| 10.4.1 指针运算符 .....             | 161 | 思考题 .....                                | 216 |
| 10.4.2 指针变量的运算 .....           | 161 | <b>第 12 章 联合、枚举、堆栈、队列<br/>与二叉树</b> ..... | 218 |
| 10.5 数组指针变量的说明和使用 .....        | 163 | 12.1 联合类型 .....                          | 218 |
| 10.6 数组名和数组指针变量作函数<br>参数 ..... | 165 | 12.1.1 基本概念 .....                        | 218 |
| 10.7 指向多维数组的指针变量 .....         | 166 | 12.1.2 定义、引用及操作 .....                    | 219 |
| 10.7.1 多维数组地址的表示方法 .....       | 166 | 12.2 枚举类型 .....                          | 221 |
| 10.7.2 多维数组的指针变量 .....         | 167 | 12.3 用 typedef 定义类型 .....                | 223 |
| 10.8 字符串指针变量的说明和使用 .....       | 167 | 12.3.1 typedef 语句的一般形式及<br>使用方法 .....    | 223 |
| 10.9 函数指针变量 .....              | 170 | 12.3.2 使用 typedef 语句应注意的<br>问题 .....     | 224 |
| 10.10 指针型函数 .....              | 171 | 12.4 堆栈与队列 .....                         | 225 |
| 10.11 指向指针的指针变量 .....          | 175 | 12.4.1 堆栈 .....                          | 226 |
| 10.12 要点 .....                 | 176 | 12.4.2 队列 .....                          | 237 |
| 思考题 .....                      | 177 | 12.5 二叉树 .....                           | 242 |
| <b>第 11 章 结构与链表</b> .....      | 179 | 12.5.1 二叉树的节点结构 .....                    | 242 |
| 11.1 结构的基本概念 .....             | 179 | 12.5.2 建立二叉树 .....                       | 247 |
| 11.2 结构类型的定义 .....             | 179 | 12.5.3 输出二叉树 .....                       | 248 |
| 11.3 结构变量的定义、引用和初始化 .....      | 181 | 思考题 .....                                | 253 |
| 11.3.1 结构变量的定义 .....           | 181 | <b>第 13 章 文件</b> .....                   | 254 |
| 11.3.2 结构变量的引用 .....           | 183 | 13.1 文件 .....                            | 254 |
| 11.3.3 结构变量的初始化 .....          | 184 | 13.1.1 基本概念 .....                        | 254 |
| 11.4 结构嵌套 .....                | 186 | 13.1.2 文件类型 .....                        | 255 |
| 11.5 结构数组 .....                | 187 | 13.1.3 文件类型的指针 .....                     | 256 |
| 11.5.1 结构数组的定义 .....           | 187 | 13.1.4 文件的打开与关闭 .....                    | 257 |
| 11.5.2 结构数组的初始化 .....          | 188 | 13.1.5 文件的读写 .....                       | 260 |
| 11.6 结构与指针 .....               | 190 | 13.1.6 文件的定位 .....                       | 269 |
| 11.6.1 结构变量指针 .....            | 190 | 13.1.7 文件检测函数 .....                      | 271 |
| 11.6.2 结构数组指针 .....            | 192 |                                          |     |
| 11.7 结构与函数 .....               | 193 |                                          |     |
| 11.7.1 结构变量作为函数参数 .....        | 193 |                                          |     |
| 11.7.2 结构变量作为函数的返回值 .....      | 195 |                                          |     |

|                                                |            |                                            |            |
|------------------------------------------------|------------|--------------------------------------------|------------|
| 13.2 非缓冲文件系统 .....                             | 272        | 15.2 Visual C++ 6.0 集成开发环境 .....           | 332        |
| 13.2.1 open 函数 .....                           | 272        | 15.2.1 Visual C++ 6.0 主界面 .....            | 332        |
| 13.2.2 close 函数 .....                          | 273        | 15.2.2 Visual C++ 6.0 帮助系统 .....           | 338        |
| 13.2.3 creat 函数 .....                          | 273        | 15.3 Visual C++ 6.0 程序的编辑及<br>调试 .....     | 339        |
| 13.2.4 read 函数 .....                           | 273        | 15.3.1 Visual C++ 6.0 应用程序的创建<br>与编辑 ..... | 339        |
| 13.2.5 write 函数 .....                          | 274        | 15.3.2 应用程序的编译、连接和运行 .....                 | 340        |
| 13.2.6 随机定位函数 .....                            | 275        | 15.3.3 程序动态调试方法 .....                      | 341        |
| 13.3 实例 .....                                  | 275        | 15.4 实例 .....                              | 343        |
| 思考题 .....                                      | 281        | 思考题 .....                                  | 344        |
| <b>第 14 章 图形与声音</b> .....                      | <b>282</b> | <b>第 16 章 C 语言程序开发实例</b> .....             | <b>345</b> |
| 14.1 图形 .....                                  | 282        | 16.1 绘制五星红旗 .....                          | 345        |
| 14.1.1 图形的基本概念 .....                           | 282        | 16.2 表盘时钟 .....                            | 348        |
| 14.1.2 图形运行程序 .....                            | 286        | 16.3 保龄球积分公告牌 .....                        | 353        |
| 14.1.3 图形模式的初始化 .....                          | 288        | 16.4 蛇形矩阵 .....                            | 356        |
| 14.1.4 图形显示方式和字符显示方式 .....                     | 292        | 16.5 国际象棋棋盘 .....                          | 358        |
| 14.1.5 基本图形函数 .....                            | 292        | 思考题 .....                                  | 359        |
| 14.1.6 图形填充函数和存取函数 .....                       | 299        | <b>第 17 章 C 语言程序开发综合案例</b> .....           | <b>360</b> |
| 14.1.7 屏幕颜色的设置和清屏函数 .....                      | 302        | 17.1 案例一：语言信号分析 .....                      | 360        |
| 14.1.8 图形窗口和图形屏幕操作函数 .....                     | 305        | 17.2 案例二：系统稳定性 .....                       | 364        |
| 14.1.9 图形模式下的文本输出 .....                        | 310        | 17.3 案例三：仪器可靠性 .....                       | 367        |
| 14.2 声音程序 .....                                | 314        | 17.4 案例四：地域导航 .....                        | 370        |
| 14.2.1 声音函数 .....                              | 314        | 17.5 案例五：地震事件检测 .....                      | 373        |
| 14.2.2 音乐 .....                                | 315        | 17.6 案例六：海啸分析 .....                        | 377        |
| 14.3 实例 .....                                  | 322        | 思考题 .....                                  | 379        |
| 思考题 .....                                      | 329        | <b>参考文献</b> .....                          | <b>380</b> |
| <b>第 15 章 Visual C++ 6.0 集成<br/>开发环境</b> ..... | <b>330</b> |                                            |            |
| 15.1 Visual C++ 6.0 的安装 .....                  | 330        |                                            |            |
| 15.1.1 Visual C++ 6.0 对系统的要求 .....             | 330        |                                            |            |
| 15.1.2 Visual C++ 6.0 的安装过程 .....              | 330        |                                            |            |

# 第 1 章 绪 论

## 【内容】

随着 C 语言的不断发展和完善,其应用日益广泛。本章从 C 语言的发展历史入手,介绍了 C 语言的各种版本以及 C 语言的特点;C 语言中的字符集、关键字以及标识符的命名;通过分析实例,总结了 C 语言的结构特点以及 C 程序的开发过程;最后详细介绍了算法的几种表达方法。

## 【目的】

初步了解 C 语言,熟悉 C 语言中的字符集、关键字以及标识符的命名方式;了解 C 语言程序的开发过程;重点掌握用流程图、N-S 图来描述算法,为后续章节的深入学习打下基础。

## 1.1 C 语言的发展

C 语言是当今社会应用广泛,并受到众多用户欢迎的一种计算机算法语言。它既可作为系统软件 的描述语言,也可用来开发应用软件。

C 语言的开发历史源于高级语言和 UNIX 操作系统的发展要求,C 语言也有一个自身的发展过程, 目前仍然处于发展和完善之中。

从历史发展来看,C 语言起源于 1963 年英国剑桥大学发表的 CPL (Combined Programming language),1967 年英国剑桥大学的 Martin Richards 对 CPL 做了简化,推出了 BCPL (Basic Combined Programming Language),1970 年美国贝尔实验室的 Ken Thompson 以 BCPL 为基础研制成了 B 语言 (取 BCPL 的第一个字母),并用 B 语言写了第一个 UNIX 操作系统,用在 PDP-7 计算机上。1973 年贝尔实验室的 D.M.Ritchie 在 B 语言的基础上研制了 C 语言,并用 C 语言写成了第一个在 PDP-11 计算机上实现的 UNIX 操作系统。1977 年出现了独立于机器的 C 语言编译文本《可移植 C 语言编译 程序》,从而大大简化了把 C 语言编译程序移植到新环境所需做的工作,这本身也就使 UNIX 操作系 统迅速地在众多的机器上实现。例如 VAX, AT&T 等计算机系统都相继开发了 UNIX。随着 UNIX 的 日益广泛使用,C 语言也迅速得到推广。

1983 年美国国家标准化协会 (ANSI) 根据 C 语言问世以来的各种版本,对 C 语言的发展和扩充 制定了新的标准,称为 ANSI C。1987 年 ANSI 又公布了新标准——87 ANSI C。1990 年,国际标 准化组织 ISO (International Standard Organization) 接受了 87 ANSI C 为 ISO C 的标准 (ISO 9899-1990)。 目前流行的 C 语言版本都是以它为基础的。

目前在微型计算机上使用的版本很多。这些不同的 C 语言版本,基本部分是相同的,但在有关 规定上又略有差异。本书以 Turbo C 2.0 的环境对 C 语言进行介绍。Turbo C 是一种快速、高效的编译 程序。它不仅提供了一个集成开发的环境,同时也按传统方式提供了一个命令行编译程序版本,以满 足不同用户的需要。

## 1.2 C 语言版本

IBM 微机 DOS、Windows 平台上常见的 C 语言版本有：

(1) Borland 公司的 Turbo C, Turbo C++, Borland C++ 及 C++ Builder (Windows 版本)。

(2) Microsoft 公司的 Microsoft C 及 Visual C++ (Windows 版本)。

这些 C 语言版本不仅实现了 ANSI C 标准, 而且在此基础上各自作了一些扩充, 使之更加方便、完美。

## 1.3 C 语言的特点

C 语言是一种高级计算机程序设计语言, 它的特点一般可归纳如下:

(1) C 语言是结构化语言。结构化语言的一个显著特点是代码和数据的分割。C 语言的主要结构成分为函数, 函数可以在程序中被定义完成独立的任务, 独立地编译成代码, 这样可以实现程序的模块化。同时, C 语言具有结构化的流程控制语句(如 if...else 语句, switch 语句, while 语句, do...while 语句, for 语句等), 支持若干种循环结构, 允许编程者采用缩进书写形式编程。因此, 用 C 语言设计出的程序层次结构清晰。

(2) C 语言运算符丰富。C 语言中运算符包含的范围很广泛。它把赋值、括号、强制类型转换都当作运算符处理。它能灵活地使用各种运算符, 可以实现在其他高级语言中难以实现的运算。

(3) C 语言数据类型丰富。数据类型有整型、实型、字符型、数组型、指针型、结构体型、共用体型等。这些数据类型能用来实现各种复杂的数据结构(如链表、树、栈等)的运算。尤其是 C 语言的指针型数据的运算, 更是灵活、多样。

(4) C 语言与计算机硬件联系紧密, 兼有高级语言与汇编语言的功能。这样可以直接访问计算机内存, 具有位操作; 生成的目标代码质量高, 运行效率远远高于一般的高级语言, 接近于汇编语言。同时 C 语言具有高级语言的可读性好、可移植性好等特点。C 语言允许直接访问物理地址, 即可直接对硬件进行操作, 实现汇编语言的大部分功能。C 语言这一特点, 使得它成为编制系统软件的基本语言(UNIX 的绝大部分就是由 C 语言写成的)。

(5) C 语言语法限制不太严格, 程序设计自由度大。例如, C 语言对数组下标越界不做检查, 整型、字符型、逻辑型数据在一定范围内可以通用, 程序书写形式较自由。这样使 C 语言能够减少对程序员的束缚。

(6) C 语言功能强大。C 语言有丰富的库函数、强大的图形处理功能, 有预处理能力, 和其他语言如汇编语言、Pascal 语言、数据库语言等的接口容易实现。C 语言程序中还可以直接调用 DOS 命令。因此, C 语言适合编写各种系统软件、工具软件等大型软件。

C 语言优点很多, 但是它也存在一些缺点, 如运算优先级太多, 数值运算能力方面不像其他高级语言那样强, 语法定义不严格等。尽管 C 语言目前还存在一些不足之处, 但由于它目标代码质量高、使用灵活、数据类型丰富、可移植性好而得到广泛的普及和迅速的发展, 成为一种受到广大用户欢迎的实用的程序设计语言, 同时也是一种在系统软件开发、科学计算、自动控制等各个领域被广泛应用的程序设计语言。

## 1.4 C 语言的基本词法

在 C 语言程序中,要用到很多概念,比如变量、函数等。为了识别这些概念,要用到一些符号来表示这些概念,标识也要有一定的规则,这就是 C 语言中的词法。它包括字符集、关键字以及标识符。

### 1.4.1 字符集

组成 C 语言源程序代码的基本字符称为 C 语言的字符集,它们是构成 C 语言的基本元素。C 语言的字符集包括:

- (1) 26 个大小写英文字母: A~Z, a~z;
- (2) 数字字符: 1, 2, 3, ..., 9;
- (3) 特殊字符: 空格、下画线 “\_” 以及 “+” “-” “\*” “/” “=” “,” “?” “;” “!” “” “%” “.” “&” “^” “#” “|” “<” “>” “|” “(” “)” “{” “}” “[” “]” “~” “\” 等。

### 1.4.2 关键字

关键字又称为保留字,它是 C 语言中预定义的单词,在程序中各自代表不同的含义。Turbo C 共有 43 个关键字,可以分成下面几类:

- (1) 描述存储属性的: auto, extern, static, register;
- (2) 描述数据类型定义的有: typedef;
- (3) 描述数据类型的有: char, int, float, double, long, short, signed, unsigned, void, struct, union, enum;
- (4) 描述语句的有: goto, if, else, switch, case, default, break, for, do, while, continue, return;
- (5) 描述访问方式的有: const, volatile;
- (6) 描述编译状态的有: sizeof;
- (7) Turbo C 扩展的关键字有: asm, \_cs, \_ds, \_es, \_ss, cdecl, far, near, huge, interrupt, pascal。

### 1.4.3 标识符

C 语言程序中的对象名是以标识符来命名的,通过标识符建立对象定义和使用的关系。例如程序中的函数、变量、数据类型、符号常量、数组、指针等都是使用标识符来命名的。通俗地讲,标识符就是名字。标识符的命名必须满足一定的构成规则。

- (1) 标识符是以字母、数字和下画线组成的,但第一个字符必须是字母或下画线。
- (2) 标识符的有效字符长度为 1~32 个字符。
- (3) 标识符中字母的大小写是有区别的,例如 NUM, num, Num 是 3 个不同的标识符。习惯上,符号常量用大写字母,变量、函数名等用小写字母,系统变量由下画线开头。

(4) 标识符不能使用 Turbo C 中的关键字, 因为关键字在系统中有特殊用途。也尽量避免使用库函数名作为标识符。“main”在 C 程序中是作为主函数名的, 因此也不应该把它作为标识符。

标识符的选择应尽量做到“见名知意”“常用取简”“专用取繁”的原则, 做到一目了然, 以增加程序的可读性。例如定义总数用“total”, 定义学号用“num”, 定义年、月、日分别用“year”“month”“day”。

## 1.5 C 程序的基本结构和结构特点

### 1.5.1 C 程序的基本结构

任何一种程序设计语言都具有特定的语法规则和规定的表达方法。一个程序只有严格按照语言规定的语法和表达方式编写, 才能保证编写的程序在计算机中能正确地执行, 同时也便于阅读和理解。

为了了解 C 语言的基本程序结构, 我们先介绍几个简单的 C 程序, 初步了解 C 语言程序的基本结构。

例 1.1 在屏幕上打印一行信息。

```
#include <stdio.h>
main()
{
    printf("This is a C program.\n");
}
```

此程序运行结果就是在屏幕上输出下面一行信息:

This is a C program.

其中 main() 表示“主函数”。每一个 C 程序都必须有一个 main() 函数。函数体由一对 {} 括起来。printf() 函数是 C 语言中的输出函数, 双引号中的字符串原样输出。“\n”是换行符, 即在输出信息后回车换行。每一语句后面都要有一分号 (;), 它是语句的组成部分, 不能遗漏。程序运行遇到“;”就认为语句结束了。

程序中的“#include <stdio.h>”是 C 预处理程序的一条指令。在编译程序之前, 凡以 # 开头的代码行都先由预处理程序预处理。该行是通知预处理程序把标准输入/输出头文件 (stdio.h) 中的内容包括到该程序中来。头文件 stdio.h 中包含了编译器在编译标准输出函数 printf 时要用到的信息和声明, 还包含了帮助编译器确定对库函数调用的编写是否正确信息。

在 C 语言程序中如果仅用到标准的输入 (scanf) / 输出 (printf) 函数时, 可以省略该行, C 语言默认包含。

例 1.2 主函数中调用子函数。

```
main()                                /*主函数*/
{
    void stac( );                      /* 函数声明 */
    int a=3;                           /*指定 a 为整数, 初始值为 3 */
}
```

```

    stac( );          /*调用函数 stac, 无返回*/
    a=endc( );        /*调用函数 endc, 结果返回给 a*/
    printf("This is a sample of c program. \n");
}
void stac( )          /*定义函数 stac, void 指定该函数不返回结果*/
{
    printf("Hello. \n");
}
int endc( )           /*定义函数 endc, int 指定该函数返回一个整数*/
{
    return (2);       /*返回整数 2*/
}

```

程序从 main( )函数处开始。变量 a 代表一个整数, 并且初始值为 3。执行程序(函数) stac( ), 屏幕上显示“Hello.”。“\n”为转义字符, 代表换行的意思。执行程序(函数) endc( ), 并将结果赋予 a, 此时, a 的值为 2。屏幕上显示“This is a sample of c program.”。

程序执行的结果是在屏幕显示两行信息:

Hello.

This is a sample of c program.

程序中的/\*...\*/表示注释部分, 注释只是为了方便阅读而加上的, 对编译和运行不起作用。注释文字可以是任意字符, 如汉字、拼音、英文等。

例 1.3 求两个整数中的大数, 并输出大数。

```

main()                /*主函数*/
{
    int a,b,c;         /*定义整型变量*/
    scanf("%d,%d",&a,&b); /*调用标准函数, 从键盘输入 a 和 b 的值*/
    c=max(a,b);        /*调用函数 max, 将返回值赋给 c*/
    printf("max=%d",c); /*输出 c 的值*/
}

int max(x,y)           /*定义 max()函数, 函数值为整型, x,y 为形式参数*/
int x,y;               /*对形参 x,y 作类型定义*/
{
    int z;             /*定义函数内部使用的变量 z*/
    if(x>y) z=x;        /*求 x,y 的最大值, 并将其赋给 z*/
    else z=y;
    return(z);         /*将 z 值返回调用处*/
}

```

程序中含有两个函数, main()为主调函数, max()为被调函数。max()的作用是将 x 和 y 中较大值

赋给 `z`。最后的 `return(z)` 是将 `z` 的值返回给主调函数。返回值是通过函数名 `max` 带回到 `main` 函数的调用处，也就是语句 “`c=max(a,b);`” `scanf()` 函数是 C 语言程序中的输入函数。在上例中的作用是输入 `a` 和 `b` 的值。`&a` 和 `&b` 中的 “`&`” 的含义是 “取地址”。“`%d,%d`” 的含义是按十进制整数形式输入。`main()` 函数在调用 `max()` 函数时将实际参数 `a` 和 `b` 的值分别传送给 `max()` 函数中的形式参数 `x` 和 `y`。函数调用后得到一个返回值赋给主函数中的变量 `c`。最后输出变量 `c` 的值等于 `a,b` 中最大的数。本程序编译后运行结果为：

```
9,7          /*这是由键盘输入的两个整数，赋给 a 和 b*/
max=9         /*程序最后的运行结果，输出给 c*/
```

从上面程序例子，可以看出 C 程序的基本结构。

C 程序为函数模块结构，所有的 C 程序都是由一个或多个函数构成，其中必须只能有一个主函数 `main()`。程序从主函数开始执行，当执行到调用函数的语句时，程序将控制转移到调用函数中执行，执行结束后，再返回主函数中继续运行，直至程序执行结束。C 程序的函数是由编译系统提供的标准函数（如 `printf`, `scanf` 等）和由用户自己定义的函数（如 `stac`, `endc` 等）。虽然从技术上讲，主函数不是 C 语言的一个成分，但它仍被看做是其中的一部分，因此，“`main`” 不能用作变量名。

函数的基本形式是：

函数类型 函数名(形式参数)

形式参数说明：

```
{
    数据说明部分；
    语句部分；
}
```

其中：

函数头：包括函数说明、函数名和圆括号中的形式参数（如 `int max(x,y)`），如果函数调用无参数传递，圆括号中形式参数为空（如 `void stac()` 函数）。

形式参数说明：指定函数调用传递参数的数据类型（如例 1.3 中语句 `int x,y;`）。

函数体：包括函数体内使用的数据说明和执行函数功能的语句，花括号 “`{`” 和 “`}`” 表示函数体的开始和结束。

## 1.5.2 C 语言的结构特点

(1) 用预处理命令 `#include` 可以包含有关文件的信息。`Turbo C` 提供了多个头文件，分类包含了各类标准函数的原型说明，需要用到某些标准库函数时必须将对应的头文件用 `#include` 语句包含在程序的首部，就可直接使用了，头文件的扩展名一般为 “.h”。

(2) C 语言程序由函数构成。每一个 C 语言程序都必须、且只能有一个 `main` 函数作为程序的主控函数，称为主函数。`main` 函数是 C 语言编译系统使用的专用名字。`main` 后面由花括号对 “`{}`” 括起来的部分是程序的主体，程序从 `main()` 函数的第一条可执行语句开始执行。“`{`” 和 “`}`” 必须成对出现，缺一不可。程序中有时会有子函数，即被主函数调用的函数。被调函数可以是系统提供的库函数，如 `printf` 和 `scanf` 函数，也可以是用户自己编写的函数，这些函数指定实际所需做的工作。如上例中的 `max` 函数。C 语言用函数来实现特定的功能，可以说 C 语言是函数式的语言，程序的全部工作都是由函数来完成的。C 语言的库函数十分丰富，这一特点使得 C 语言容易实现程序模块化。

一个较完整的 C 语言程序大致包括:

头文件(一组#include语句, 也称包含文件)

用户函数说明部分

·全程变量定义

主函数

若干用户自己编写的函数

在主函数和其他函数中一般又包含局部变量定义、若干个用户函数的调用语句等。

(3) “/\*...\*/”之间的内容构成 C 语言程序的注释部分。“/\*...\*/”之间的内容可以是一行, 也可以是多行。注释部分不参与程序的编译和执行, 只是起说明作用, 增加程序的可读性。/\*和\*/必须成对出现, 且/与\*之间不允许有其他字符(如空格)出现。通常注释可以出现在程序段前面, 以对该程序加以说明, 或者在语句行的后面出现, 以对该语句加以说明。

(4) 大小写字母在 C 语言中是有区别的。如“Main”“MAIN”“main”“MAIN”将被 C 语言视为不同。Turbo C 所理解的程序主函数名称为“main”, 这点应引起读者注意。

(5) C 程序书写格式自由, 一行可以写一条语句, 也可以写多条语句, 一条语句也可以分成几行写, 没有行号, 只要在语句结束后加上分号即可。C 语言规定可以在任何一个分隔符和空格处换行。

(6) 每条语句以分号结束, 分号是语句的组成部分, 即使是最后一条语句, 分号也不可缺少。

## 1.6 C 语言程序开发过程

### 1. 编辑

C 源程序可以用任何编辑程序来输入, 如 Word、WPS、记事本等, 或在 C 的集成开发环境下编辑, C 源程序通常以.c 作为扩展名。

### 2. 编译和连接

C 语言源程序通过编辑方式输入后, 如果要执行的话, 则必须先进行编译, 编译后生成.obj 文件。在主菜单下选择 **compile** 项, 回车, 在其下拉菜单中选择子项“**compile to OBJ**”, 按回车键, 即进入编译过程生成目标文件。然后再选择 **compile** 下拉菜单子项 “**Make ExE File**”, 按回车键, 文件进行连接并生成可执行文件.exe。

若程序有错误, 编译将会产生警告信息并在屏幕底部信息窗口显示出错信息。按 **Alt+E** 可重新回到编辑窗口对程序进行修改, 然后再编译连接, 直至程序可以正确运行为止。

源程序编译连接成功后, 便可执行该程序。

程序执行有如下几种方式:

(1) 按 **F10** 键激活主菜单, 并用光标移动键将亮条移至 **Run** 项回车, 再在下拉子菜单内选择 **Run** 后回车, 程序便开始执行。

(2) 直接按 **F9** 键运行程序。程序运行后, 若想看到运行结果, 可选择 **Run** 子菜单中的 **User Screen** 项, 便可切换到用户屏幕观看运行结果。也可直接按 **Alt+F5** 键切换到用户屏幕, 程序执行的结果显示在用户屏幕上, 看完后可按任意键回到 TC 环境。

(3) 在 DOS 状态下运行已连接生成的扩展名为 .exe 的文件。如果用户认为自己的源程序不会有编译、连接错误时, 在源程序编辑完成后, 就可直接用 **RUN** 命令或直接按 **Ctrl+F9** 键。这时 Turbo

C 将一次完成从编译、连接到运行的全过程。

### 3. 执行和调试

编译完成后, 用 RUN 命令或直接按 Ctrl+F9 键, 执行形成的 .exe 文件。

整个 C 源程序的编辑、编译、运行、调试的过程可以用图 1.1 表示。

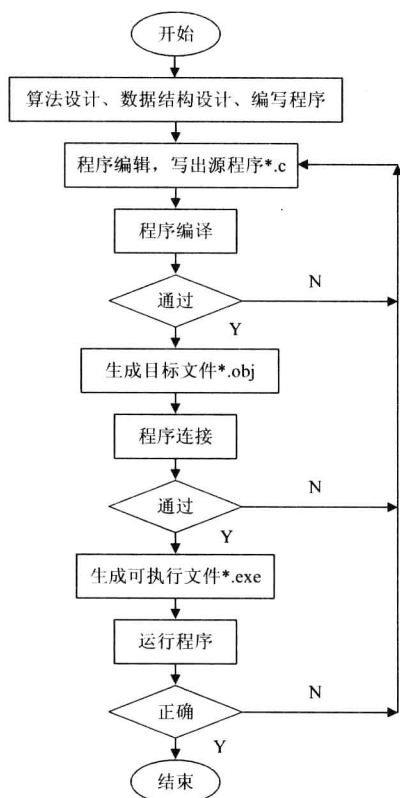


图 1.1 C 语言程序的开发过程

## 1.7 C 语言程序开发环境

编写好的 C 语言程序要经过编辑(输入)、编译和连接后才能形成可执行的程序。图 1.2 表示 C 语言程序设计上机过程。



图 1.2 C 语言程序上机过程

C 语言的编译系统有多种, 本书采用 Turbo C 2.0 集成开发环境。Turbo C 2.0 不仅是一个快速、高效的编译程序, 同时还带有一个易学、易用的集成开发环境。使用 Turbo C 2.0 无须独立地编辑、编译和连接程序, 就能建立并运行 C 语言程序, 因为, 这些功能都组合在 Turbo C 2.0 的集成开发环境内, 并且可以通过一个简单清晰的主屏幕使用这些功能。Turbo C 2.0 的集成开发环境将在第 2 章

给出详细的介绍。

## 1.8 算 法

著名计算机科学家沃思提出一个公式：

$$\text{数据结构} + \text{算法} = \text{程序}$$

数据结构就是在程序中指定数据的类型和数据的组织形式，算法就是操作步骤。对于面向对象程序设计，强调的是数据结构，而对于面向过程的程序设计语言如 C, Pascal, FORTRAN 等语言，主要关注的是算法。掌握算法，也是为面向对象程序设计打下一个扎实的基础。那么，什么是算法呢？

### 1.8.1 算法的概念和特性

人们使用计算机，就是要利用计算机处理各种不同的问题，而要做到这一点，人们就必须事先对各类问题进行分析，确定解决问题的具体方法和步骤，再编制好一组让计算机执行的指令即程序，交给计算机，让计算机按人们指定的步骤有效地工作。这些具体的方法和步骤，其实就是解决一个问题的算法。根据算法，依据某种规则编写计算机执行的命令序列，就是编制程序，而书写时所应遵守的规则，即为某种语言的语法。

从广义上讲，算法就是为解决一个问题而采取的方法和步骤。我们要编写一 C 语言程序，必须考虑如何做才能达到预期的结果。程序中的操作语句，就是算法的体现。

算法是要经过设计的。对于同一个问题，有多种不同的解题方法和步骤。有的方法只需进行很少的步骤，而有些方法则需要较多的步骤。一般来说，我们希望采用简单的和运算步骤少的方法。因此，为了有效地进行解题，不仅需要保证算法正确，还要考虑算法的质量，选择合适的算法。

算法可分为两大类：数值运算算法和非数值运算算法。数值运算是求数值解，例如求圆面积，求方程的根等，都属于数值运算范围。非数值运算包括图书检索、人事管理，公交调度等问题。

下面先来看几个简单算法的例子。

例 1.4 有 50 个考生，要求将他们之中成绩在 80 分以上者打印出来。用  $n$  表示考生学号， $n_1$  代表第 1 个考生学号， $n_i$  代表第  $i$  个考生学号。用  $g_i$  代表考生成绩， $g_i$  代表第  $i$  个考生成绩，算法可表示如下：

step1: 将 1 赋值给  $i$ ;

step2: 如果  $g_i \geq 80$ ，则打印  $n_i$  和  $g_i$ ，否则不打印；

step3: 将  $i+1$  赋值给  $i$ ;

step4: 如果  $i \leq 50$ ，返回到 step2，继续执行，否则算法结束。

用计算机实现循环运算是轻而易举的事情，所有的计算机高级语言中都有实现循环的语句。循环语句很好地体现了计算机高效、准确的计算能力，是计算机能实现的较好的算法。

例 1.5 判断一个大于或等于 3 的正整数是否是一个素数。

分析：所谓素数，是指除了 1 和该数本身之外，不能被其他任何整数整除的数。判断一个数（用  $n$  表示）是否为素数的方法：将数  $n$  作为被除数，将 2 到  $(n+1)$  各个整数轮流作为除数，如果都不能被整除，则  $n$  为素数。

算法：设要判断是否为素数的数为  $n$ 。