21 世 纪 高 等 院 校 计 算 机 规 划 教 材

C语言
程序设计教程

主　编　周鸣争
副主编　钟志水　苏守宝
主　审　孙家启

电子科技大学出版社

21 世纪高等院校计算机规划教材

C 语言程序设计教程

主　编　周鸣争

副主编　钟志水　苏守宝

编　写　钟志水　刘　涛　许　斗　苏守宝
　　汪　伟　汪朝霞　周鸣争

主　审　孙家启

电子科技大学出版社

内容简介

本书全面地介绍了C语言的基本概念,C语言的数据类型与运算规则,C语言语句及结构特点;系统地阐述了C语言程序设计的基本方法和技巧;对面向对象程序设计及C++的编程技术也作了初步的介绍;同时,兼顾覆盖了全国计算机等级考试以及全国高等学校(安徽考区)计算机基础教育、教学(考试)新大纲的内容。本书可作为普通高等学校和高职高专计算机程序设计课程的教材,也可作为成人高等教育的培训教材及广大科技工作者的自学和考试参考书。

图书在版编目(CIP)数据

C语言程序设计教程/周鸣争主编.—成都:电子科技大学出版社,2005.8
ISBN 7-81094-901-2

Ⅰ.C… Ⅱ.周… 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(2005)第094899号

C语言程序设计教程　　周鸣争　主编

出　　版:电子科技大学出版社(成都建设北路二段四号　邮政编码:610054)
责任编辑:张　俊
发　　行:电子科技大学出版社
印　　刷:中国科技大学印刷厂
开　　本:787mm×1092mm　1/16　　印张　20.375　字数　470千
版　　次:2005年8月第1版
印　　次:2005年8第1次印刷
书　　号:ISBN 7-81094-901-2/TP·471
定　　价:28.00元

前　言

C语言是近年来应用较广的一种现代编译型语言，它既具有多种高级语言的优点，又具有低级语言的功能。C语言功能丰富、表达力强、运算速度快、目标效率高、可移植性好，而且可以直接实现对系统硬件接口的控制，具有较强的系统处理能力。尤其是C++的出现，进一步增加了C语言面向对象的程序设计功能，使得C语言已成为最流行的一种计算机程序设计语言，并作为目前大多数高等学校理工科专业学生的必修或选修课程。为了使初学程序设计的读者能够掌握C语言的程序设计方法，并初步具备使用C语言开发应用程序和解决实际问题的能力，我们根据国家教育部全国高等学校计算机基础课程教学指导委员会制定的C语言程序设计教学要求，同时兼顾全国高等学校(安徽考区)计算机基础教育、教学(考试)新大纲的内容，结合我们近几年"C语言程序设计"精品课程建设的教学实践，确定了本书的组织与结构。

本书共分11章，全面地介绍了C语言的基本概念、C语言的数据类型与运算规则、C语言语句及结构特点；系统地阐述了C语言程序设计的基本方法和技巧；对面向对象程序设计及C++的编程技术也作了初步的介绍。

本书内容丰富，坚持理论联系实践、深入浅出、循序渐进的原则；在强调基本概念的基础上，引入了大量的实例来阐明各种应用问题。本书注重培养读者的程序设计能力以及良好的程序设计风格；力求使读者通过学习，能够对C语言编程方法及程序设计技术有一个较为全面的了解，为进一步提高计算机应用软件设计与开发打下坚实的基础。另外，每章含有大量习题和上机实验内容，方便读者自测和上机实践。

本书由周鸣争教授主编，钟志水、苏守宝担任副主编，由安徽省高等学校计算机基础课程教学指导委员会副主任兼秘书长孙家启教授主审。其中第1、6章由钟志水编写，第2章由刘涛编写，第3、4章由许斗编写，第5章由苏守宝编写，第7、8章由汪伟编写，第9、10章由汪朝霞编写，第11章及附录由周鸣争编写。全书由周鸣争负责统稿和定稿工作。参与本书编写和统稿工作的还有刘涛、强俊、楚宁、夏跃武等。

在本书的编写过程中得到了有关专家热心的指导与无私的帮助，电子科技大学出版社为本书的尽快出版做了大量的工作，编者在此一并表示衷心的感谢。此外，本书写作时还参考了大量文献资料，在此向这些文献资料的作者深表谢意。

本书可作为普通高等学校和高职高专计算机程序设计课程的教材，也可作为成人高等教育的培训教材及广大科技工作者的自学和考试参考书。

由于时间仓促和水平所限，书中难免有不当和欠妥之处，敬请各位专家、读者不吝批评指正。

编　者

2005 年 5 月

目　录

第 1 章　C 语言概述 …… 1

1.1　C 语言的发展及特点 …… 1
1.1.1　C 语言的发展过程 …… 1
1.1.2　C 语言的特点 …… 1
1.2　C 语言程序的基本结构 …… 2
1.3　算法及其描述 …… 5
1.3.1　算法的概念 …… 5
1.3.2　算法的特性 …… 5
1.3.3　简单算法举例 …… 6
1.3.4　算法的描述 …… 7
1.4　C 语言字符集、标识符与关键字 …… 15
1.4.1　C 语言字符集 …… 15
1.4.2　标识符 …… 16
1.4.3　关键字 …… 16
1.5　C 语言程序的上机步骤 …… 17
1.5.1　上机步骤 …… 17
1.5.2　Turbo C 2.0 介绍 …… 17
习　题 …… 20

第 2 章　数据类型与表达式 …… 21

2.1　C 语言的数据类型 …… 21
2.2　常量与变量 …… 22
2.2.1　常量和符号常量 …… 22
2.2.2　变量 …… 23
2.3　整型数据 …… 23
2.3.1　整型常量的表示方法 …… 23
2.3.2　整型变量 …… 24
2.4　实型数据 …… 27
2.4.1　实型常量的表示方法 …… 27
2.4.2　实型变量 …… 27

2.5 字符型数据 …… 28
2.5.1 字符常量 …… 28
2.5.2 字符变量 …… 30
2.5.3 字符数据在内存中的存储形式及其使用方法 …… 30
2.5.4 字符串常量 …… 31
2.6 运算符和表达式 …… 32
2.6.1 C语言运算符与表达式简介 …… 32
2.6.2 算术运算符和算术表达式 …… 32
2.6.3 赋值运算符和赋值表达式 …… 35
2.6.4 逗号运算符和逗号表达式 …… 36
2.6.5 条件运算符 …… 37
2.7 不同类型数据间的混合运算 …… 37
习 题 …… 40

第3章 顺序程序设计 …… 43

3.1 C语言的基本语句 …… 43
3.2 数据输入与输出 …… 44
3.2.1 数据输入输出的概念 …… 44
3.2.2 字符数据的输入与输出 …… 44
3.2.3 格式化输入与输出 …… 46
3.3 顺序结构程序设计举例 …… 53
习 题 …… 54

第4章 选择与循环结构程序设计 …… 58

4.1 选择结构(分支结构)程序 …… 58
4.1.1 关系运算符与关系表达式 …… 58
4.1.2 逻辑运算符与逻辑表达式 …… 59
4.1.3 if语句 …… 60
4.1.4 switch语句 …… 64
4.1.5 程序举例 …… 66
4.2 循环结构程序 …… 69
4.2.1 while语句 …… 69
4.2.2 do-while语句 …… 70
4.2.3 for语句 …… 70
4.2.4 转移语句 …… 73
4.2.5 循环的嵌套 …… 75
4.2.6 程序举例 …… 76
习 题 …… 78

第5章 数 组 …… 85
5.1 一维数组的定义和引用 …… 85
5.1.1 一维数组的定义 …… 85
5.1.2 一维数组元素的引用 …… 86
5.1.3 一维数组的初始化 …… 86
5.1.4 一维数组程序举例 …… 87
5.2 二维数组的定义和引用 …… 89
5.2.1 二维数组的定义 …… 89
5.2.2 二维数组元素的引用 …… 90
5.2.3 二维数组的初始化 …… 90
5.2.4 二组数组程序举例 …… 91
5.3 字符数组 …… 93
5.3.1 字符数组的定义 …… 93
5.3.2 字符数组的初始化 …… 94
5.3.3 字符串的输入与输出 …… 95
5.3.4 字符串处理函数 …… 96
5.3.5 字符数组应用举例 …… 99
习 题 …… 101
第6章 函数与编译预处理 …… 105
6.1 概述 …… 105
6.2 函数的定义 …… 106
6.2.1 无参函数的定义 …… 106
6.2.2 有参函数的定义 …… 107
6.2.3 空函数 …… 107
6.2.4 形参和实参 …… 107
6.2.5 函数的返回值 …… 108
6.3 函数的调用 …… 110
6.3.1 函数的调用 …… 110
6.3.2 函数调用的方式 …… 111
6.3.3 对被调用函数的声明和函数原型 …… 111
6.4 函数的嵌套和递归调用 …… 113
6.4.1 函数的嵌套调用 …… 113
6.4.2 函数的递归调用 …… 115
6.5 数组作为函数参数 …… 117
6.5.1 数组元素作为函数实参 …… 117
6.5.2 数组名作为函数参数 …… 117

6.5.3 用二维数组名作函数参数 …… 120
6.6 变量的作用域 …… 120
6.6.1 局部变量 …… 120
6.6.2 全局变量 …… 121
6.7 变量的存储方式 …… 123
6.7.1 动态存储方式与静态存储方式 …… 123
6.7.2 自动变量 …… 123
6.7.3 静态局部变量 …… 124
6.7.4 寄存器变量 …… 125
6.7.5 用 extern 声明外部变量 …… 126
6.7.6 用 static 声明外部变量 …… 128
6.7.7 关于变量的声明和定义 …… 128
6.8 内部函数和外部函数 …… 129
6.8.1 内部函数 …… 129
6.8.2 外部函数 …… 130
6.9 编译预处理 …… 131
6.9.1 宏定义 …… 132
6.9.2 “文件包含”处理 …… 135
6.9.3 条件编译 …… 137
6.10 程序举例 …… 139
习 题 …… 144

第7章 指 针 …… 149

7.1 指针与指针变量 …… 149
7.1.1 内存单元、地址和指针 …… 149
7.1.2 指针变量的定义、赋值与引用 …… 151
7.2 指针变量的运算 …… 153
7.3 指针与数组 …… 155
7.3.1 指针与一维数组 …… 155
7.3.2 指针与多维数组 …… 158
7.4 指针与函数 …… 160
7.4.1 指针作为函数的参数 …… 160
7.4.2 指向函数的指针 …… 161
7.4.3 指针型函数 …… 163
7.5 指针与字符串 …… 165
7.5.1 字符串的表达形式 …… 165
7.5.2 字符指针作为函数参数 …… 166
7.5.3 使用字符指针与字符数组的区别 …… 167

7.6 指针数组与命令行参数 …… 168
7.6.1 指针数组的定义和使用 …… 168
7.6.2 指向指针的指针 …… 170
7.6.3 指针数组作 main 函数的命令行参数 …… 172
7.7 程序举例 …… 173
习 题 …… 178

第 8 章 结构体与共用体 …… 184

8.1 结构体类型的定义 …… 184
8.2 结构体类型变量的定义、引用和初始化 …… 185
8.2.1 结构体类型变量的定义 …… 185
8.2.2 结构体类型变量的引用及初始化 …… 186
8.3 结构体数组 …… 188
8.4 结构体指针变量 …… 191
8.4.1 结构体指针变量的说明和使用 …… 191
8.4.2 结构体数组指针变量 …… 193
8.4.3 结构体指针变量作函数参数 …… 194
8.5 链表 …… 196
8.5.1 链表的概述 …… 196
8.5.2 创建并输出单链表 …… 199
8.5.3 单链表的删除和插入 …… 200
8.6 共用体、枚举和用户自定义类型 …… 204
8.6.1 共用体类型 …… 204
8.6.2 枚举类型 …… 207
8.6.3 用户自定义类型 …… 209
8.7 程序举例 …… 210
习 题 …… 213

第 9 章 位运算 …… 218

9.1 位运算的概念 …… 218
9.1.1 计算机内数据的表示方法 …… 218
9.1.2 位运算及其运算符 …… 219
9.2 位运算 …… 219
9.2.1 按位与 …… 219
9.2.2 按位或 …… 220
9.2.3 按位异或 …… 220
9.2.4 按位取反 …… 221
9.2.5 按位左移 …… 221

9.2.6 按位右移 …… 222
9.2.7 位运算赋值运算 …… 223
9.3 位段简介 …… 223
9.3.1 位段的概念与定义 …… 223
9.3.2 关于位段的定义和引用的几点说明 …… 224
习 题 …… 225

第10章 文 件 …… 227

10.1 C语言文件概述 …… 227
10.1.1 文件与文件名 …… 227
10.1.2 文件分类 …… 227
10.1.3 缓冲文件系统 …… 228
10.2 文件类型指针 …… 229
10.3 文件的打开与关闭 …… 229
10.3.1 文件的打开 …… 229
10.3.2 文件的关闭 …… 231
10.4 文件的读写操作 …… 232
10.4.1 检测文件是否结束函数 …… 232
10.4.2 字符读写函数 …… 232
10.4.3 字符串读写函数 …… 234
10.4.4 数据块读写函数 …… 236
10.4.5 格式化读写函数 …… 238
10.5 位置指针与文件定位 …… 238
10.5.1 位置指针复位函数 …… 238
10.5.2 随机读写与 fseek()函数 …… 239
10.5.3 返回文件当前位置的函数 …… 240
10.6 文件的操作状态和出错检测 …… 241
10.6.1 ferror()函数 …… 241
10.6.2 clearerr()函数 …… 242
习 题 …… 242

第11章 C++程序设计基础 …… 244

11.1 C++概述 …… 244
11.1.1 面向对象的程序设计 …… 244
11.1.2 C++语言的发展及特点 …… 246
11.2 C++程序的基本结构 …… 247
11.3 C++对C基本功能的扩充 …… 248
11.3.1 C++中的关键字 …… 248

11.3.2 函数声明 …… 248
11.3.3 函数的重载 …… 248
11.3.4 灵活的变量说明 …… 249
11.3.5 作用域标识符 …… 249
11.3.6 C++中扩充的基本功能 …… 249
11.4 C++的类和对象 …… 250
11.4.1 类 …… 251
11.4.2 对象 …… 255
11.5 构造函数与析构函数 …… 257
11.5.1 构造函数 …… 257
11.5.2 成员初始化表 …… 258
11.5.3 析构函数 …… 259
11.5.4 自引用指针 this …… 262
11.5.5 重载函数 …… 263
11.5.6 友元函数 …… 264
11.6 继承与派生类 …… 266
11.6.1 派生类声明 …… 267
11.6.2 派生类的构造函数 …… 268
11.7 多态性与虚函数 …… 270
11.7.1 静态联编与动态联编 …… 270
11.7.2 虚函数 …… 270
11.7.3 纯虚函数与抽象类 …… 274
习 题 …… 276

附录Ⅰ 常用字符与 ASCII 代码对照表 …… 279
附录Ⅱ C 语言运算符的优先级与结合性 …… 280
附录Ⅲ C 语言常用库函数 …… 282
附录Ⅳ C 语言常见错误信息表 …… 291
附录Ⅴ C 语言上机实验指导 …… 295
参考文献 …… 313

第 1 章

C 语言概述

C 语言是国际上广泛流行的计算机高级语言。本章主要阐述 C 语言的发展过程及特点；C 程序的基本结构、算法及其描述；C 语言字符集、标识符与关键字以及 C 程序的上机步骤，并介绍 C 语言程序上机步骤及开发环境。

1.1 C 语言的发展及特点

1.1.1 C 语言的发展过程

C 语言自流行以来一直兴盛不衰，也是目前应用最广泛的计算机语言之一。因为它既具有高级语言的优点，又具有低级语言的许多特点，因此它既可作为系统软件描述语言，也可用来编写应用软件。

C 语言是在 20 世纪 70 年代初问世的。1978 年美国电话电报公司(AT&T)贝尔实验室正式发表了 C 语言。与此同时，B. W. Kernighan 和 D. M. Ritchit(合称 K&R)合著了著名的《The C Programming Language》一书，在该书中并没有定义一个完整的标准 C 语言，后来由美国国家标准协会(American National Standards Institute)在此基础上制定了一个 C 语言标准，于 1983 年发表，通常称之为 ANSI C。

早期的 C 语言主要是用于 UNIX 系统。由于 C 语言的强大功能和各方面的优点逐渐为人们认识，到了 20 世纪 80 年代，C 语言开始进入其他操作系统，并很快在各类大、中、小和微型计算机上得到了广泛的使用，成为当代最优秀的程序设计语言之一。

K&R 在 1988 年修改了他们的经典著作《The C Programming Language》，按照 ANSI C标准重写了该书。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 87 ANSI C 为 ISO C 的标准(ISO 9899－1990)。目前流行的 C 语言编译系统都是以它为基础的，本书的叙述基本上以 ANSI C 为基础。目前广泛流行的各种版本 C 语言编译系统虽然基本部分是相同的，但也有一些不同。在微型机上使用的有 Microsoft C、Turbo C、Quick C、Borland C 等，它们的不同版本又略有差异。因此，读者应了解自己使用的计算机系统所配置的 C 语言编译系统的特点和规定。

1.1.2 C 语言的特点

C 语言的主要特点有：

(1)语言简洁、紧凑,使用方便、灵活。C语言共有32个关键字(见本章1.4节),9种控制语句,程序书写形式自由。不像Fortran语言,对程序的书写格式严格限制。

(2)运算符丰富。C语言的运算符包含的范围很广泛,共有34种运算符(参见附录Ⅱ)。C语言把括号、赋值、强制类型转换等作为运算符处理,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语言中难以实现的运算。

(3)具有丰富的数据类型。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的数据结构(如链表、树、栈等)的运算,尤其是指针类型数据,使用起来更为灵活、多样。

(4)具有结构化的控制语句(如if…else语句、while语句、do…while语句、switch语句、for语句)。用函数作为程序的模块单位,便于实现程序的模块化。C语言是理想的结构化语言,符合现代编程风格的要求。

(5)语法限制不太严格,程序设计自由度大。例如对数组下标越界不做检查,由程序编写者自己保证程序的正确。对变量的类型使用比较灵活,例如整型数据、字符型数据、逻辑型数据可以通用。

(6)C语言允许直接访问物理地址,能进行位(bit)操作,能实现汇编语言的大部分功能,可以直接访问硬件。因此,有人把C语言称为"中级语言",意为兼有高级语言和低级语言的特点。

(7)生成目标代码质量高,程序执行效率高。一般只比汇编程序生成的目标代码效率低10%~20%。

(8)用C语言写的程序可移植性好(与汇编语言比)。基本上不做修改就能用于各种型号的计算机和各种操作系统。

以上只是从C语言的整体角度出发,介绍了C语言与其他语言相比较的一般特点,至于C语言内部的其他特点将结合以后各章的内容作详细介绍。

1.2 C语言程序的基本结构

在学习C语言的具体语法之前,我们先通过几个简单的C程序示例,初步了解C语言程序的基本结构和特性。

【例1.1】 编写显示字符串"Hello,World!"的程序。

```
main ( )
{
  printf ("Hello,World! \n");
}
```

本程序的作用是输出以下一行信息:

```
Hello,World!
```

其中main是"主函数"名,每一个C语言程序都必须有一个main函数。其后的一对圆括号是函数的标志,圆括号中间可以是空的,但这一对圆括号不能省略。函数体由大括

弧{}括起来。

本例中主函数内只有一个输出语句，printf 是 C 语言中的输出函数(详见第 3 章)。双引号(双括号)内的字符串原样输出。"\n"是换行符，即在输出"Hello, World!"后回车换行，语句最后有一分号。

【例 1.2】 编写程序，实现求两整数之差。

```
main ( )                    /*求两数之差*/
{ int a,b,result;           /*定义变量*/
  a=345; b=112;             /* 以下3行为C语句*/
  result=a-b;
  printf ("result is %d\n",result);
}
```

本程序的作用是求两个整数 a 和 b 之差 result。/*……*/表示注释部分，为便于理解，可用汉字表示注释，当然也可以用英语或汉字拼音作注释。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。第 2 行是声明部分，定义变量 a 和 b，指定 a 和 b 为整型(int)变量。第 3 行是两个赋值语句，使 a 和 b 的值分别为 345 和 112。第 4 行使 result 的值为 a-b。第 5 行中"%d"是输入输出的"格式字符串"，用来指定输入输出时的数据类型和格式(详见第 3 章)，"%d"表示"以十进制整数形式输出"。在执行输出时，此位置上代以一个十进制整数值。printf 函数中括弧内最右端 result 是要输出的变量，现在它的值为 233(即 345-112 的值)。因此输出一行信息为：

```
result is 233
```

【例 1.3】 编写一函数，实现求两个整数之和，并在 main 函数中调用。

```
main ( )                          /* 主函数*/
{ int a,b,c;                      /*声明部分，定义变量*/
  scanf ("%d,%d",&a,&b);          /*输入变量a和b的值*/
  c=sum(a,b);                     /*调用sum函数，将得到的值赋给c*/
  printf ("sum=%d\n",c);          /*输出c的值*/
}
int sum(int x,int y)   /*定义sum函数，函数值为整型，形式参数x,y为整型*/
{ int z;
  z=x+y;
  return (z);    /*将z的值返回，通过sum带回调用处*/
}
```

本程序包括两个函数：主函数 main 和被调用的函数 sum。sum 函数的作用是将 x、y 之和的值赋给变量 z，return 语句将 z 的值返回给主调函数 main，返回值是通过函数名 sum 带回到 main 函数的调用处。

main 函数中的 scanf 是"输入函数"的名字(scanf 和 printf 都是 C 系统提供的标准输入输出函数)。程序中 scanf 函数的作用是输入 a 和 b 的值。&a 和 &b 中的"&"的含义是"取地址"，此 scanf 函数的作用是将两个数值分别输入到变量 a 和 b 的地址所标志的

单元中，也就是输入给变量 a 和 b。&a 和 &b 前面的“%d，%d”的含义与前面相同，只是现在用于“输入”。它指定输入的两个数据按十进制整数形式输入。关于 scanf 函数详见第 3 章。

main 函数中第 4 行为调用 sum 函数，在调用时将实际参数 a 和 b 的值分别传送给 sum 函数中的形式参数 x 和 y。经过执行 sum 函数得到一个返回值（即 sum 函数中变量 z 的值），把这个值赋给变量 c，然后输出 c 的值。printf 函数中双引号内的“sum＝%d”，在输出时，其中“%d”将由 c 的值取代之，“sum ＝”原样输出。程序运行情况如下：

```
451,234 ↵      (输入 451 和 234 给 a 和 b)
sum=685        (输出 c 的值)
```

通过以上几个例子，我们可以看出 C 语言程序的基本结构和常见特性如下：

(1) C 程序是由函数构成的。一个 C 源程序至少包含一个 main 函数，也可以包含一个 main 函数和若干个其他函数，因此，函数是 C 程序的基本单位。被调用的函数可以是系统提供的库函数（例如 printf 和 scanf 函数），也可以是用户根据需要自己编制设计的函数（例如，例 1.3 中的 sum 函数），其调用方式是一样的。

(2) 一个函数由两部分组成：

① 函数头或函数首部，即函数的第一行。包括函数名、函数类型、函数参数（形参）名、参数类型。如果函数没有参数，圆括号中形式参数为空。

例如，例 1.3 中的 sum 函数的首部为

```
int      sum      (int      x,      int      y)
 ↓        ↓        ↓        ↓        ↓        ↓
函数类型  函数名  函数参数类型  函数参数名  函数参数类型  函数参数名
```

② 函数体，即函数首部下面的大括弧{……}内的部分。函数体一般包括：

声明部分：定义该函数所用到的变量，如例 1.3 中 main 函数中的“int a,b,c;”和“int z;”。在第 6 章中还将会看到，可在声明部分对所调用的函数进行声明。

执行语句部分：为完成函数功能所需要的若干语句。

(3) C 程序书写格式自由，一行内可以写几个语句，一个语句可以分写在多行上。C 程序没有行号。

(4) 每个语句和数据定义的最后必须有一个分号，分号是 C 语句的必要组成部分。例如：

```
c=a+b;
```

但是函数首部后面不能跟分号。

(5) 可以用/*……*/对 C 程序中的任何部分作注释。一个好的、有使用价值的源程序都应当加上必要的注释，以增加程序的可读性。

(6) 一个 C 程序总是从 main 函数开始执行的，而不论 main 函数在整个程序中的位置如何（main 函数可以放在程序最前面，也可以放在程序最后，或在一些函数之前，在另一些函数之后）。

1.3 算法及其描述

1.3.1 算法的概念

解决任何一个问题都需要有算法，所谓算法就是为解决一个问题而采取的方法和步骤。当然，任何问题中都会有各种各样的数据，算法就是按一定的次序对这些数据进行处理。数据的类型和组织形式(即数据结构)不同，对应的算法也不同。

数据是操作的对象，算法的目的是对数据进行加工处理以得到期望结果。打个比方：现在我们要请客吃饭，正确的做法是：先买菜，然后洗菜，最后是炒菜。如果你是把菜买回来后，先炒再洗，那就是算法错了。还有一点也很重要，那就是我们在设计算法时一定要站在计算机这个角度来考虑，否则再好的算法也只能放弃。比如说，现在想在计算机上打印一张九九乘法表，如果我们对计算机说：打印一张九九乘法表。这个算法很好，但目前的计算机没办法实现，所以这个算法是无效的。从上面的分析可以看出，程序设计时必须认真考虑和设计数据结构以及算法。因此，著名计算机科学家沃思(Nikiklaus Wirth)提出了程序定义的著名公式：

程序＝数据结构＋算法

这个公式的重要性在于它说明了程序与算法的关系，同时说明了数据结构的选择也是十分重要的。对于程序而言，算法和数据结构是统一的关系。

算法是指对特定问题求解步骤的一种描述，同一个问题可以有不同的算法。例如，求1＋2＋3＋…＋100。有人可能先进行1＋2，再加3，再加4，一直加到100。而有的人采取这样的算法：100＋(1＋99)＋(2＋98)＋…＋(49＋51)＋50＝100＋49×100＋50＝5050。还可以有其他的算法。当然，算法有优劣之分，有的算法只需进行很少的步骤，而有些算法则需要较多的步骤。一般来说，希望采用简单的和运算步骤少的算法。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。

计算机算法可分为两大类别：数值运算算法和非数值运算算法。数值运算的目的是要求数值解，例如求方程的根、求一个函数的定积分等，都属于数值运算范围。非数值运算包括的面十分广泛，最常见的是用于事务管理领域，例如图书检索、信息查询、人事管理、行车调度管理等。

1.3.2 算法的特性

一个好的算法一般应具有以下特点：

(1)有穷性。一个算法只能包含有限步的操作，而不能是无限的，它必须在有限次执行后完成。这里的“有限”是指“在合理的范围之内有限”。如果让计算机执行一个历时1000年才结束的算法，这虽然是有限的，但超过了合理的限度，人们不可能等那么长时间，所以这个算法不能算有效算法。

(2)确定性。算法中的每一个步骤都应当是确定的，而不应当是含糊的、模棱两可的。算法中的每一个步骤应当不致被解释成不同的含义，而应是十分明确无误的。如“n被一