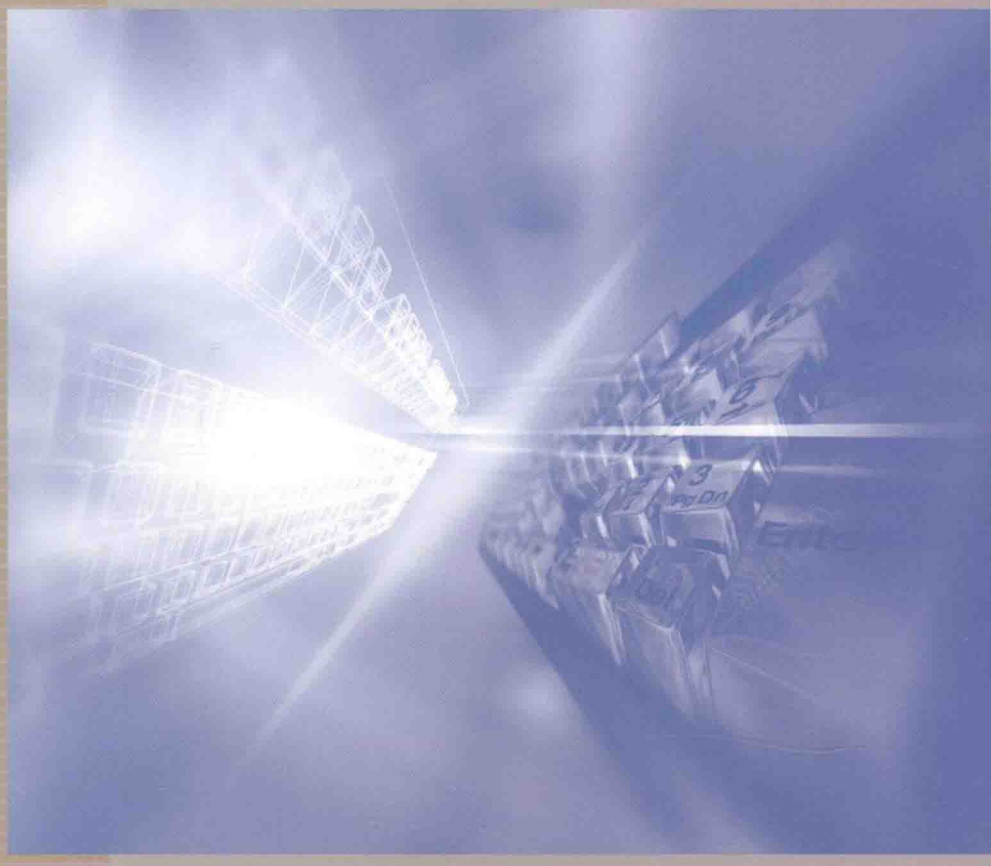
高等职业教育“十一五”规划教材 • 计算机信息类

C语言程序设计

C YUYAN CHENGXU SHEJI

张利群　苏金芝　王巍慈　编著



中国石化出版社
HTTP://WWW.SINOPEC-PRESS.COM

高等职业教育“十一五”规划教材·计算机信息类

C语言程序设计

张利群　苏金芝　王巍慈　编著

中国石化出版社

内 容 提 要

本书所介绍的C语言是目前最流行的程序设计语言。全书共12章，详细介绍了C语言的基本概念、数据类型及运算；利用控制流程语句、数组、函数、结构体、指针以及文件等进行C语言程序设计的方法；在Visual C++6.0环境下建立、编辑、编译、连接和运行C语言程序的方法。本书包含了大量编著者的程序设计思想和经验；例题典型、丰富，有一定深度；每章都精选了习题，仔细编排了实训内容。全书深入浅出，重点突出，循序渐进，实用性强。

本书可作为大中专院校《C语言程序设计》课程教材和计算机培训教材，也可以作为全国计算机等级考试二级教材。

图书在版编目(CIP)数据

C语言程序设计／张利群，苏金芝，王巍慈编著．
—北京：中国石化出版社，2010.8
ISBN 978-7-5114-0441-1

Ⅰ.①C… Ⅱ.①张… ②苏… ③王… Ⅲ.①C语言－程序设计 Ⅳ.①TP312

中国版本图书馆CIP数据核字(2010)第128867号

中国石化出版社出版发行
地址：北京市东城区安定门外大街58号
邮编：100011　电话：(010)84271850
读者服务部电话：(010)84289974
http://www.sinopec-press.com
E-mail：press@sinopec.com.cn
河北天普润印刷厂印刷
全国各地新华书店经销
*
787×1092毫米 16开本 14.75印张 363千字
2010年8月第1版　2010年8月第1次印刷
定价：28.00元

前　言

C 语言是目前最流行的程序设计语言，它功能丰富，表达能力强，使用灵活方便，可移植性强，代码质量高，集高级语言和低级语言的优点于一身，是一门中级语言，广泛适用于系统程序和应用程序的设计。

近年来，C 语言在系统软件、工具软件、图像处理、人工智能、数据处理等方面都得到了广泛的应用。《C 语言程序设计》课程已作为计算机专业和非计算机专业所开设的计算机语言课程的首选课程。

针对初学者的特点，本书对内容进行了精心安排，用读者易理解的方法去组织教材、介绍知识，全书共分 12 章：第 1 章介绍了 C 语言的特点、程序的构成、算法的概念和特性等；第 2 章介绍了 C 语言的数据类型及运算等；第 3 章介绍了顺序结构程序设计方法；第 4 章介绍了选择结构程序设计方法；第 5 章介绍了循环结构程序设计方法；第 6 章介绍了数组的应用方法；第 7 章介绍了函数的应用方法及变量的种类等；第 8 章介绍了编译预处理；第 9 章介绍了结构体、共用体与枚举类型数据的使用方法等；第 10 章介绍了指针的应用方法；第 11 章介绍了位运算；第 12 章介绍了文件的应用。

本书具有如下几个显著特点：

1. 内容安排合理，各章衔接得好，以 C 语言的最基本内容为主线，深入浅出，通俗易懂，把难点进行了分散，使初学者更好地掌握课程的基本内容。

2. 本书以全国计算机等级考试二级 C 语言考试大纲为主要编写依据，精选了教学内容，使教材更适合于学生参加计算机等级考试的需求。

3. 在各章中，均包含了大量编著者的程序设计思想、程序设计理念和程序设计方法，使读者少走弯路，尽快掌握应用 C 语言进行程序设计的思想和方法。

4. 典型例题，有示范作用；精选习题，有利于读者掌握知识点和考点；实训内容丰富，有利于培养读者的调试程序能力，进而全面掌握 C 语言的知识。

5. 注重实训，书中实训内容丰富、选题恰当，给出了在 Visual C++6.0 环境下建立、编译、连接和运行 C 语言程序的方法，全书所有程序均在此环境下调试通过并正确运行。

本书由辽宁石油化工大学张利群(第 7 章、第 9 章、第 10 章、第 12 章以及附录)、苏金芝(第 1 章、第 2 章、第 3 章、第 4 章)、王巍慈(第 5 章、第 6 章、第 8 章、第 11 章)编著完成。

杨治东、张实、吉书朋、于永安、石丽君、徐晓军对本书的出版、编写提供了许多帮助，在此对他们表示诚挚的谢意！

由于作者水平有限，书中难免有错误和疏漏，恳请读者批评指正。

目　录

第1章　C语言概述 …… (1)
1.1　C语言的特点 …… (1)
1.2　C程序的构成 …… (1)
1.2.1　简单的C源程序举例 …… (1)
1.2.2　C程序的基本结构 …… (3)
1.3　算法的概念和特性 …… (4)
1.3.1　算法的概念 …… (4)
1.3.2　算法的特征 …… (4)
1.3.3　算法的复杂度 …… (4)
1.3.4　算法的设计原则 …… (5)
1.4　算法的表示 …… (5)
1.4.1　伪代码表示算法 …… (5)
1.4.2　流程图表示算法 …… (7)
1.4.3　N-S流程图表示算法 …… (8)
1.5　结构化程序设计 …… (11)
习题1 …… (11)
第2章　数据类型及其运算 …… (14)
2.1　C语言的数据类型 …… (14)
2.2　常量和变量 …… (14)
2.2.1　常量 …… (14)
2.2.2　变量 …… (15)
2.3　整数类型 …… (16)
2.3.1　整型常量的表示方法 …… (16)
2.3.2　整型变量 …… (16)
2.4　实数类型 …… (17)
2.4.1　实型常量的表示方法 …… (17)
2.4.2　实型变量 …… (17)
2.5　字符型数据 …… (18)
2.5.1　字符型常量 …… (18)
2.5.2　字符型变量 …… (19)
2.5.3　字符串常量 …… (19)
2.6　算术运算符和算术表达式 …… (19)
2.6.1　算术运算符 …… (19)
2.6.2　算术表达式 …… (21)
2.7　赋值运算符和赋值表达式 …… (22)

2.7.1 简单赋值表达式 …… (22)
2.7.2 复合赋值表达式 …… (22)
2.8 逗号运算符和逗号表达式 …… (23)
2.9 关系运算符和关系表达式 …… (24)
2.10 逻辑运算符和逻辑表达式 …… (24)
2.11 各种数值型数据间的混合运算 …… (26)
2.12 运算优先级和结合性 …… (26)
习题2 …… (27)
实训1 C语言的上机环境 …… (29)
第3章 顺序结构程序设计 …… (32)
3.1 赋值语句 …… (32)
3.2 数据输入输出 …… (32)
3.3 格式输入与输出 …… (33)
3.3.1 格式输入 …… (33)
3.3.2 格式输出 …… (34)
3.4 专用于字符数据的输入与输出函数 …… (36)
3.4.1 getchar()函数 …… (36)
3.4.2 putchar()函数 …… (37)
3.5 应用举例 …… (37)
习题3 …… (39)
实训2 顺序结构程序设计 …… (41)
第4章 选择结构程序设计 …… (42)
4.1 if语句 …… (42)
4.1.1 if语句的格式 …… (42)
4.1.2 if语句的嵌套 …… (45)
4.1.3 条件运算符 …… (46)
4.2 switch语句 …… (47)
4.3 应用举例 …… (48)
习题4 …… (52)
实训3 选择结构程序设计 …… (54)
第5章 循环结构程序设计 …… (57)
5.1 for语句 …… (57)
5.2 while语句 …… (58)
5.3 do~while语句 …… (61)
5.4 循环的嵌套 …… (63)
5.5 break和continue语句 …… (64)
5.5.1 break语句 …… (64)
5.5.2 continue语句 …… (66)
5.6 goto语句 …… (67)
5.7 应用举例 …… (67)

习题5 …… (70)
实训4 循环结构程序设计 …… (74)
第6章 数组 …… (76)
6.1 一维数组的定义和引用 …… (76)
6.1.1 一维数组的定义 …… (76)
6.1.2 一维数组的初始化 …… (76)
6.1.3 一维数组元素的引用 …… (77)
6.1.4 一维数组的应用举例 …… (78)
6.2 二维数组的定义和引用 …… (81)
6.2.1 二维数组的定义 …… (81)
6.2.2 二维数组的初始化 …… (81)
6.2.3 二维数组元素的引用 …… (82)
6.2.4 二维数组的应用举例 …… (83)
6.2.5 大于二维的数组定义和引用 …… (84)
6.3 字符串和字符数组 …… (85)
6.3.1 字符串和字符串结束标记 …… (85)
6.3.2 字符数组的定义 …… (85)
6.3.3 字符数组的初始化 …… (85)
6.3.4 字符数组的输入输出 …… (87)
6.3.5 字符串处理函数 …… (88)
6.3.6 字符数组应用举例 …… (90)
习题6 …… (92)
实训5 数组的应用 …… (96)
第7章 函数 …… (98)
7.1 库函数的调用 …… (98)
7.2 函数定义的一般形式 …… (99)
7.3 函数的参数和返回值 …… (100)
7.3.1 形式参数与实在参数 …… (100)
7.3.2 函数的返回值 …… (101)
7.4 函数的调用 …… (101)
7.4.1 函数调用的一般形式 …… (101)
7.4.2 调用方式 …… (104)
7.5 函数的嵌套调用 …… (104)
7.6 函数的递归调用 …… (107)
7.7 参数传递的方式 …… (110)
7.8 局部变量和全局变量 …… (112)
7.8.1 局部变量 …… (112)
7.8.2 全局变量 …… (113)
7.9 变量的存储类型 …… (114)
7.9.1 自动变量 …… (115)

7.9.2 寄存器变量 …… (116)
7.9.3 静态变量 …… (117)
7.9.4 外部变量 …… (118)
7.10 内外部函数和多文件的程序 …… (118)
7.10.1 内部、外部函数 …… (118)
7.10.2 多文件的程序 …… (119)
习题7 …… (120)
实训6 递归程序设计 …… (124)
第8章 编译预处理 …… (127)
8.1 宏定义和调用 …… (127)
8.2 文件包含处理 …… (130)
习题8 …… (130)
第9章 结构体、共用体与枚举类型 …… (134)
9.1 结构体 …… (134)
9.1.1 结构体类型变量的定义 …… (134)
9.1.2 结构体类型变量的初始化 …… (136)
9.1.3 结构体类型变量的引用 …… (136)
9.1.4 结构体数组 …… (137)
9.1.5 应用举例 …… (139)
9.2 共用体 …… (141)
9.2.1 共用体类型变量的定义 …… (141)
9.2.2 共用体类型变量的引用 …… (142)
9.3 枚举类型 …… (144)
9.4 用 typedef 说明数据类型 …… (146)
习题9 …… (146)
实训7 结构体类型程序设计 …… (150)
第10章 指针 …… (153)
10.1 地址和指针的概念 …… (153)
10.2 指针变量的定义 …… (153)
10.3 指针变量的引用 …… (154)
10.4 指针变量作为函数参数 …… (156)
10.5 指针与数组 …… (159)
10.5.1 指针与一维数组 …… (159)
10.5.2 指针与二维数组 …… (161)
10.6 指针与字符串 …… (163)
10.7 指向函数的指针 …… (165)
10.8 返回指针值的函数 …… (166)
10.9 指针数组和指向指针的指针 …… (167)
10.9.1 指针数组 …… (167)
10.9.2 指向指针的指针 …… (170)

10.10 指向结构体类型数据的指针 …… (171)
10.10.1 指向结构体变量的指针 …… (171)
10.10.2 链表的建立 …… (172)
10.10.3 对链表的处理 …… (175)
习题10 …… (177)
实训8 指针的应用 …… (181)
第11章 位运算 …… (183)
11.1 位运算符和位运算 …… (183)
11.2 应用举例 …… (186)
习题11 …… (187)
第12章 文件 …… (190)
12.1 文件类型指针 …… (190)
12.2 文件的打开和关闭 …… (190)
12.2.1 文件的打开 …… (191)
12.2.2 文件的关闭 …… (192)
12.3 文件的读写 …… (192)
12.3.1 fputc 函数和 fgetc 函数 …… (192)
12.3.2 fputs 函数和 fgets 函数 …… (194)
12.3.3 fread 函数和 fwrite 函数 …… (196)
12.3.4 fprintf 函数和 fscanf 函数 …… (198)
12.4 文件的定位 …… (200)
12.4.1 rewind 函数 …… (200)
12.4.2 fseek 函数 …… (201)
12.4.3 ftell 函数 …… (202)
12.5 应用举例 …… (202)
习题12 …… (205)
实训9 文件的使用 …… (208)
附录1 C语言的上机环境介绍 …… (212)
附录2 C语言的关键字 …… (219)
附录3 C语言的常用库函数 …… (221)
附录4 常用字符与ASCII码对照表 …… (224)
参考文献 …… (225)

第1章　C语言概述

计算机在程序的控制下完成各种任务。程序是软件开发人员根据用户需求开发的、用程序设计语言描述的、适合计算机执行的指令序列。程序设计就是编程人员根据要处理问题的解决步骤，按一定的逻辑关系，将一系列的计算机指令组合在一起，最终使问题得到解决的过程。C语言是常用的程序设计语言之一，人们借助C语言已经开发出了大量的系统程序和应用程序。

1.1　C语言的特点

19世纪70年代，贝尔实验室的D. M. Ritchie在B语言基础上推出了C语言。几十年来，它迅速发展，成为最受欢迎、应用最广泛的计算机语言之一。C语言之所以具有这么强的生命力，最主要的原因是它具有不同于其他语言的优势。

C语言具有下列特点：

1. C语言语法结构简单

C语言以接近英语国家的自然语言和数学语言作为语言的表达形式。它语言简洁、紧凑、使用方便灵活，压缩了所有不必要的成分，只有32个关键字和9种控制语句。程序书写格式自由，一个语句可以分写在多行上，一行内也可以书写多个语句。

2. C语言数据类型和运算符丰富

C语言具有各种数据类型，能实现各种复杂的数据结构的运算，特别是C语言引入了指针概念，可使程序效率更高。另外C语言具有强大的图形功能，支持多种显示器和驱动器。C语言的运算符也相当丰富，共有34种，灵活地使用这些运算符可以方便实现各种运算。

3. C语言是模块化、结构化程序设计语言

C语言用函数作为程序模块以实现程序的模块化。C语言具有多种循环、条件语句，控制程序的流向，使程序结构化。C语言是完全模块化、结构化的语言。

4. C语言提供了丰富的标准库函数

C语言提供了大量的标准库函数。库函数越多，需要用户自己编制的程序就越少。C语言中所有的输入输出操作都是通过库函数实现的。标准库中的函数非常丰富、灵活，这使得用C语言编程非常方便。

1.2　C程序的构成

1.2.1　简单的C源程序举例

下面通过几个简单的C程序例子，说明C语言源程序的书写规则和基本结构。

例1.1　输出一行文字。

```
#include<stdio. h>
void main()                              /*主函数*/
```

```
{
  printf("This is my first C program.");          /*输出函数调用*/
}
```

该程序的运行结果是输出一行字符：

This is my first C program.

其中 main() 表示“主函数”，每一个 C 程序都必须有且只能有一个主函数。main 前面的 void 表示这个“主函数”是“空类型”，即执行完这个函数后不产生函数值。由花括号“{ }”括起来的是函数体，printf() 是输出函数。printf() 函数中双引号内的字符串原样输出。/* …… */表示注释部分，用于解释该程序或该语句的作用。注释对系统编译和运行不起任何作用，经常出现在被注释语句的后面或前面。

程序中使用了标准函数库中的 printf() 函数。C 语言规定，程序使用标准库函数时，应提供这些函数的相关信息，即出处。程序第一行的“#include<stdio.h>”的作用就是提供 printf() 函数相关信息的，它来自于 stdio.h 头文件。

例 1.2 计算两个整数的平方和的平方根。

```
#include<stdio.h>                                 /*指明本程序包含 stdio.h 头文件*/
#include<math.h>                                  /*指明本程序包含 math.h 头文件*/
void main()
{
  int i,j;                                        /*声明 i,j 两个整型变量*/
  float k;                                        /*声明一个实型变量 k */
  i=3;j=5;                                        /*给 i,j 赋值*/
  k=sqrt(i*i+j*j);                                /*计算并赋值*/
  printf("i=%d,j=%d,k=%f\n",i,j,k);               /*输出 i,j 及 k 值*/
}
```

该程序的运行结果：

i=3,j=5,k=5.830952

本程序的作用是求两个整数 i 和 j 的平方和的平方根。其中，“i=%d,j=%d,k=%f\n”是输出的“格式控制字符串”，“\n”表示换行。sqrt() 函数的作用是求平方根，它是数学函数库中的函数，必须在程序开头加“#include<math.h>”命令，即把头文件“math.h”包含到文件中。

例 1.3 求两个数的最大值。

```
#include<stdio.h>
int max(int,int);                                 /*使用函数的声明*/
void main()
{
  int a,b,m;                                      /*定义变量 a 和 b */
  scanf("%d,%d",&a,&b);                           /*从键盘输入 a 和 b 的值*/
  m=max(a,b);
  /*调用 max 函数,并将 a 和 b 的值对应传给 x 和 y,将结果赋给 m 变量*/
  printf("max=%d\n",m);                           /*输出 m 的值*/
```

```
}
int max(int x,int y)                    /*函数首部*/
/*定义 max 函数,函数返回值为 int 型,两个形式参数 x、y 均为 int 型*/
{
  int n;                                /*定义 max 函数中的变量 n */
  if(x>y)
    n=x;          /*条件判断语句,如果 x>y 成立,则将 x 的值赋给变量 n */
  else
    n=y;          /*如果 x>y 不成立,则将 y 的值赋给变量 n */
  return n;       /*将 n 值从 max 函数带回到主函数*/
}
```

运行程序时，先在键盘输入：

10,20↙　　（输入 10 和 20 分别赋值给变量 a 和变量 b）

屏幕将显示：max=20

本程序包括两个函数：主函数 main()和被调用的函数 max()。max()函数的作用是将变量 x 和 y 中较大者的值赋给变量 n，然后由 return 语句将 n 的值返回给主调函数 main()。返回值是通过函数名 max 带回到 main()函数的调用处。

程序的第 2 行是对 max()函数的声明，max()函数的位置如果在调用函数之后，为了让编译系统正确识别和调用 max()函数，必须在调用 max()函数之前对其进行声明。因此，在 C 语言程序设计中，要遵循先声明后使用原则。scanf()函数的作用是从键盘输入变量 a 和 b 的值。

1.2.2　C 程序的基本结构

函数是构成 C 程序的基本单位。一个 C 程序由一个或多个函数组成，一个 C 函数是实现一个特定功能的程序段，它一般由若干条 C 语句组成。C 语句是完成程序功能的最小单位。

C 程序的基本构成：

（1）C 程序是由函数构成的。一个 C 程序有且只有一个主函数 main()，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标准库函数，也可以是用户自己编写的函数。

（2）一个函数由两部分组成，即函数的首部和函数体。

① 函数的首部，即函数的第一行。包括函数属性、函数名、函数参数(形参)名、参数类型等。

例如，例 1.3 中的 max()函数的首部为：

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

在 C 语言中，函数有两个属性：函数类型和函数的存储类别。函数类型指函数返回值的类型；函数的存储类别指的是函数能否被其他源文件调用，具体内容将在第 7 章讲述。一个函数名后面必须跟一对圆括号，函数参数可以没有，如 main()，但圆括号不能省略。

② 函数体，即函数首部下面的花括号“{ }”内的部分。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。

函数体一般包括以下两部分：声明部分和执行部分。声明部分用于声明变量和声明调用的函数，执行部分是由若干条语句组成的，用以实现该函数的功能。

（3）一个 C 程序的执行总是从 main()开始，再由 main()结束。

（4）C 程序中一行内可以写几个语句，语句间用分号隔开。一个语句也可以分写在多行上，写在多行上的一个语句用花括号“{ }”括起来。

（5）C 语言本身没有输入输出语句，它的输入输出操作是由标准库函数完成的。

（6）可以用/* …… */(注意 /与 * 之间不能有空格)对 C 程序中的任何部分作注释。

（7）C 语言中的每个语句和数据声明都要以分号结束，分号是 C 语句的必要组成部分。

1.3 算法的概念和特性

采用结构化程序设计方法进行程序设计，一般包括四个方面的内容：数据结构(对数据的描述)、算法(对操作的描述)、程序设计方法和语言工具。其中，算法是灵魂，数据结构是加工对象，语言是编程工具。程序设计的关键之一，是算法，即解题的方法和步骤。

1.3.1 算法的概念

所谓算法是指为解决某一个问题而采取的有效的、科学的方法和步骤。我们使用计算机解决某个问题时，如果能够在有限的存储空间内运行有限个程序语句而得到正确的结果，则称这个算法是可用的。但必须注意的是，算法不等于程序，也不是计算方法。程序员可以使用任何一种计算机语言将算法转换成程序。

1.3.2 算法的特征

解决同一个问题，可以采用不同的算法。为了更有效地应用计算机资源，我们不仅需要保证算法正确，还应考虑算法的质量。一个算法应具有以下 5 个特性：

（1）有穷性。一个算法的有穷性，是指算法必须能在合理的执行时间内执行有限个步骤之后结束。

（2）确定性。算法的确定性，是指算法中的每一个步骤必须有明确的定义，不允许存在二义性。

（3）可行性。算法的可行性，是指算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

（4）有零个或多个输入。算法的运算对象是数据，而数据大多数情况下需要通过输入得到。

（5）有一个或多个输出。算法的目的是为了求问题的“解”，这些问题的“解”需要通过输出得到。

1.3.3 算法的复杂度

评价一个算法的主要标准是算法的执行效率和存储需求。通常使用时间复杂度和空间复杂度来衡量这两个标准。

算法的时间复杂度是指执行算法所需的计算工作量。为了能够客观地衡量一个算法的执行效率，不受所使用计算机硬件、程序设计语言和算法实现过程中的细节影响，一般使用算法在执行过程中所需基本运算的执行次数来度量算法的时间复杂度。

空间复杂度用来度量算法所需存储空间的大小。一个算法所占的存储空间包括算法程序所占的空间、输入数据所占的空间以及算法执行过程中所需的额外空间。

1.3.4 算法的设计原则

一个算法在设计过程中应遵循以下 4 个原则：

（1）正确性。算法的正确性指算法没有错误，并且对不同的输入数据都能得到正确的输出结果。

（2）可读性。一个好的算法应该是方便人阅读与交流的，避免使用晦涩难懂的算法。

（3）健壮性。算法的健壮性指算法处理非法数据的能力，当算法被输入非法数据时，算法能适当地作出反应或进行处理。

（4）高效率与低存储量需求。效率指的是算法的执行时间，若同一个问题可以使用多种算法解决，执行时间短的算法执行效率高；存储量需求是指一个算法在执行过程中所需要的最大存储空间，低存储量需求的算法为好。

1.4 算法的表示

最常用的算法表示法是伪代码表示法和流程图表示法，其中流程图表示法又分为传统流程图和 N－S 图两种。

1.4.1 伪代码表示算法

伪代码表示法是介于自然语言和计算机语言之间的一种不受语法约束的语言描述方式。用伪代码描述算法，就像写文章一样，把算法步骤自上而下写下来，每一行或几行表示一个操作。它不使用图形符号，书写方便，格式紧凑，简便易懂，可以方便地转换成计算机编程语言。

例 1.4 求 n!。

用伪代码表示的算法如下：

```
start
将 f 的值赋为 1
将 temp 的值赋为 2
while temp≤n do
  f⇐f * temp
  temp⇐temp + 1
end do
输出 f 的值
end
```

例 1.5 判断大于 2 的整数 n 是否是素数。

用伪代码表示的算法如下：

```
start
将 i 的值赋为 2
while i≤√n do
  r⇐mod(n,i)
```

```
    if r=0 then
      n⇐i
    else
      i⇐i+1
    end if
end do
if  i=n then
    输出 n 不是素数
else
    输出 n 为素数
end if
end
```

例 1.6 将 1900 年至 2000 年间的闰年打印出来。

用伪代码表示的算法如下：

```
start
将 y 赋值为 1900
while y≤2000 do
if mod(y,4)=0 then
  if mod(y,100)≠0 then
    print:y:"是闰年"
  else
    if mod(y,400)=0 then
      print:y:"是闰年"
    else
      print:y:"不是闰年"
    end if
  end if
else
    print:y:"不是闰年"
end if
y⇐y+1
end do
end
```

例 1.7 求 $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots+\frac{1}{21}$。

用伪代码表示的算法如下：

```
start
sum⇐1
i⇐3
sign⇐1
```

```
while i≤21 do
  sign⇐sign * (-1)
  sum⇐sum + sign * 1/i
  i⇐i +2
end do
print sum
end
```

通过上面的例题，我们可以看出伪代码书写格式自由，容易修改。使用伪代码，可以帮助我们更好地表述算法，不用拘泥于具体的实现。使用伪代码的目的是为了使被描述的算法可以容易地用任何一种编程语言(Pascal、C、Java)实现。因此，使用伪代码表示法结构清晰、代码简单、可读性好，并且类似自然语言。

1.4.2 流程图表示算法

流程图由一些图框和流程线组成，其中图框表示各种操作的类型，图框中的文字和符号表示操作的内容，流程线表示操作的先后次序。使用图形表示算法的思路是一种极好的方法，因为千言万语不如一张图。它直观形象，易于理解。

例 1.8 依次输入 10 个数，要求打印输出最小的数。

解：流程图见图 1-1。

例 1.9 有 3 个数，把它们按从小到大的顺序打印出来。

解：流程图见图 1-2。

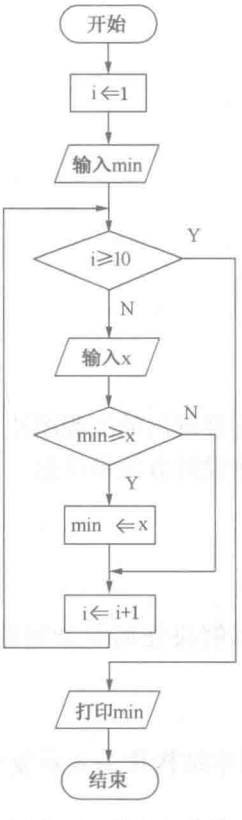


图 1-1 输出最小数

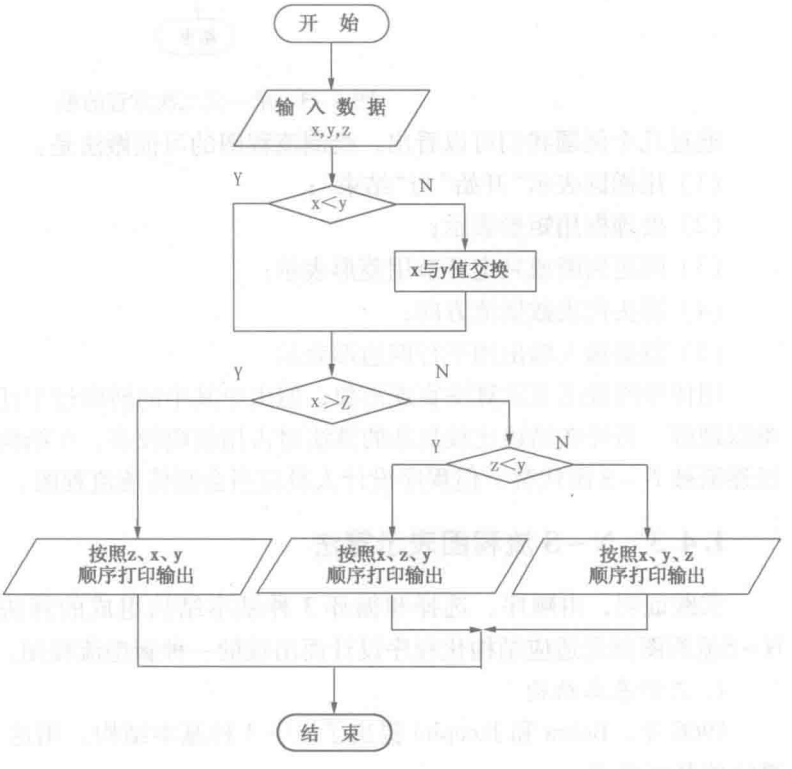


图 1-2 将 3 个数从小到大排序

例 1.10 求方程式 $ax^2+bx+c=0$ 的根。分别考虑有两个不等实根、两个相等实根和两个不等虚根三种情况。

解：流程图见图 1－3。

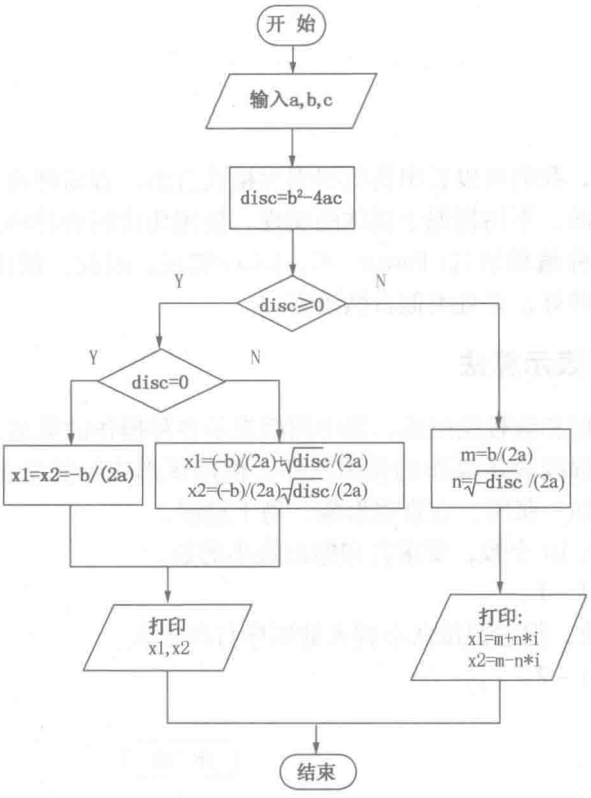


图 1－3 求一元二次方程的根

通过几个例题我们可以看出，绘制流程图的习惯做法是：

（1）用椭圆表示“开始”与“结束”；

（2）处理框用矩形表示；

（3）问题判断或判定环节用菱形表示；

（4）箭头代表数据流方向；

（5）数据输入输出用平行四边形表示。

用传统流程图表示算法直观形象，但由于其中的转向过于任意，这样画出的流程图让人难以理解，另外在描述比较复杂的算法时占用篇幅较多，在结构化程序设计方法推出后，已经逐渐被 N－S 图代替，但程序设计人员应当会画传统流程图。

1.4.3 N－S 流程图表示算法

实践证明，由顺序、选择和循环 3 种基本结构组成的算法，可以解决任何复杂问题。N－S流程图就是适应结构化程序设计而出现的一种新型流程图。

1. 三种基本结构

1966 年，Bohra 和 Jacopini 提出了以下 3 种基本结构，用这 3 种基本结构作为表示复杂算法的基本单元。