高职高专成人高等教育教材

程序设计方法教程

陈士龙　郭炜懋　编著

電子工業出版社

高职高专成人高等教育教材

程序设计方法教程

陈士龙　郭炜懋　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内容简介

本书采用结构化程序设计方法,“自顶向下,逐步求精”地进行问题分析和程序设计。全书共分9章。第1章介绍了程序设计方法、算法概念以及设计框架,给出了结构化程序设计的基本思路和表达工具;第2章至第5章详细介绍了应用软件的屏幕界面设计,举例说明了设计的实现,给出了功能独立且专一的模块程序;第6章至第8章介绍了管理软件设计中常用的经典算法,结合实例讨论了这些算法的应用;第9章综合前面各章基本知识,设计了有一定难度和实用价值的管理系统,给出了全部功能的源程序。每章配有综合实例、小结和习题。

本书可作高等职业学校、高等专科学校和成人高等教育的教材,也可供其他计算机技术人员选用。

图书在版编目(CIP)数据

程序设计方法教程/陈士龙,郭炜懋编著 . - 北京:电子工业出版社,2000.6
高职高专成人高等教育教材
ISBN 7-5053-5875-8

Ⅰ.程… Ⅱ.①陈… ②郭… Ⅲ.程序设计-方法-高等学校-教材 Ⅳ.TP311.11

中国版本图书馆 CIP 数据核字(2000)第 04205 号

丛 书 名:高职高专成人高等教育教材
书　　名:**程序设计方法教程**
编 著 者:陈士龙　郭炜懋
策划编辑:卢先河
责任编辑:刘文杰
特约编辑:张国栋
排版制作:电子工业出版社计算机排版室
印 刷 者:北京李史山胶印厂
出版发行:电子工业出版社　URL:http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编 100036
经　　销:各地新华书店
开　　本:787×1092　1/16　印张:13.5　字数:340 千字
版　　次:2000 年 6 月第 1 版　2000 年 6 月第 1 次印刷
书　　号:ISBN 7-5053-5875-8/G·535
印　　数:8000 册　　定价:20.00 元

前　言

程序设计的实质问题就是把人工求解问题的过程转换为计算机的算法语言源程序，计算机执行该程序，获得问题的解或指出问题无解。

求解一个问题，首先应分析问题本身，描述它的数据结构，书写问题的求解步骤，这是算法设计阶段；然后，选择某种算法语言（如 C 语言），运用结构化程序设计原则，表达上述算法，这是程序设计阶段。

因此，不管是用计算机做科学计算还是企事业管理，任何问题的计算和求解，算法和方法是相辅相承的，其中算法设计占有特别重要的位置。

本书采用结构化程序设计方法，基于模块化（面向过程）程序设计思想，“自顶向下，逐步求精”地进行问题分析和程序设计。书的主体是讨论管理领域的数据处理，研究常规问题的算法设计，给出算法的 C 语言实现。

学习这本书的最终目标，使学生学会设计有一定实用性的管理软件，如学生成绩管理、库存物品管理、职工档案管理、乘坐飞机的旅客信息登计管理等。

为解决上述问题，总结近年来的教学经验，书的编写遵循理论联系实际的宗旨，内容由浅入深，突出要点，讲解详细和通俗易懂，每章都配有综合实例样品，并且前后章节之间衔接紧密。

本书共分 9 章，第 1 章介绍了程序设计方法、算法概念以及设计框架，给出了结构化程序设计的基本思路和表达工具；第 2 章至第 5 章详细介绍了应用软件的屏幕界面设计，如键盘、光标、视屏、鼠标操作和菜单、窗口设计，举例说明了设计的实现，给出了功能独立且专一的模块程序；第 6 章至第 8 章介绍了管理软件设计中常用的经典算法，如数据编辑、查询、排序和链表运算，结合实例讨论了这些算法的应用；第 9 章是综合应用前面章节的基本知识，设计了有一定难度和实用价值的学生成绩管理系统，给出了全部功能的源程序。

全书的 C 程序，在 UCDOS 环境下，用 turbo_C 系统调试和运行。由于水平所限，加之时间仓促，错误和疏漏之处难免，敬请读者批评指正。

编　者

2000 年 3 月

第1章　方法和算法概述

程序设计的实质问题就是把人工求解问题的过程转换为计算机的算法语言源程序，计算机执行该程序，获得问题的解或指出问题无解。

求解一个问题，首先应分析问题本身，描述它的数据结构，书写问题的求解步骤，这是算法设计阶段；然后，选择某种算法语言（如C语言），运用结构化程序设计原则，表达上述算法，这是程序设计阶段。

因此，不管是用计算机做科学计算还是企事业管理，任何问题的计算和求解，算法和方法是相辅相承的，其中“算法设计”占有特别重要的位置。

本章将扼要讨论：结构化程序设计方法，结构化程序的基本结构，用结构化方法做程序的总体设计和详细设计的基本要点；算法和算法的特性，算法的描述，算法的C语言表示，模块化程序设计的基本准则。

1.1　方法

目前，程序设计方法有两大类，一个是面向过程的模块化程序设计方法，另一个是面向对象的程序设计方法。本书基于模块化程序设计思想，采用“自顶向下，逐步求精”方法，进行问题分析和程序设计。

1.1.1　结构化程序设计方法

结构化程序设计方法是基于模块化的编程思想，对实际问题进行分析和编程时，先从总体出法，把一个大问题或较复杂问题分解为若干个相对独立的子问题，对每个子问题可以再细化为若干个低一层的子问题，直到子问题便于在计算机上实现，这就是自顶向下、逐步求精的解题方法。

在计算机上实现各个子问题的求解，就是设计相应功能的子程序，即功能模块。这些模块，功能独立且专一，便于用相关算法描述之。当用C语言来表达算法时，就是功能独立的C函数。因此，结构化程序设计方法，仍是程序员作软件设计时普遍采用的方法。下面，用这种方法求解了一个实际问题，演示了分析求解的全过程。

【例1.1】设计一个程序，对输入的一行正文，先分离出其中的英文单词，再统计包含元音的单词个数和包含四个不同元音的单词个数。

例如，若输入的文本行是：program main(input,output)；应取出四个英文单词，其中包含元音的单词是四个，包含四个不同元音的单词为0个。

解题思路：若输入的文本在buf中，从buf中分离出一个英文单词时，先确定单词首字母，然后顺序取字母且放入数组word内，遇到非字母字符时，则该单词已拼成。重复上述操作，直到buf中文本被取完。若数组word内共放进Na个单词，再测试每个单词内是否包含元音和是否包含四个不同元音，测试时，用Nb计数包含元音的单词数，用Nc计数包含四个不同元音的

单词数。

假设输入的文本是小写的字母，buf 和 word 的说明为：

```
#define N   50
#define M   15
char   buf[80],Word[N][M];
int    Na,Nb,Nc;
```

1. 总体框架分析

(1) 将待测文本输入 buf。

(2) 从 buf 中分解出 Na 个单词存入 Word。

(3) 检测 Word 中单词，统计出包含元音的单词 Nb 个、包含四个不同元音的单词 Nc 个。

2. 细化总体框架

(1) 从 buf 中分解出 Na 个单词的操作可细化为：

```
令 Na = 0, pos = 0 ;
while( buf[pos] 不是尾标 )
   { 搜索 buf,确定第 Na 个单词首字母位置 pos;
     依据 pos,从 buf 中取出第 Na 个单词存入 Word;
   }
```

(2) 测试 Word 中元音单词和四个不同元音单词的操作可细化为：

```
for( i = 1; i < = Na; i++ )
{
   测试第 Word[i]个单词是否包含元音,若是则 Nb 计数一次;
}
for( i = 1; i < = Na; i++ )
{
   测试第 Word[i]个单词是否包含四个不同元音,若是则 Nc 计数一次;
}
```

3. 模块化

函数 ScanPos()，确定 buf 中某个单词首字母位置 pos，或者文本已分析完。

```
ScanPos( char buf[ ],int *pos )
   {
     while( ! ( buf[*pos] > = 'a' && buf[*pos] < = 'z' ) && buf[*pos] ! = '\0' )
       (*pos)++;
   }
```

函数 GetWord()，依据 pos 从 buf 中取字符拼第 Na 个单词，且存入 Word。

```
GetWord( char buf[ ],int *pos )
   { int j;
     j = 0; Na++;
```

```
    while( buf[*pos] >='a' && buf[*pos] <='z' )
      { Word[Na][j]= buf[*pos];
        j++, (*pos)++ ;
      }
  }
```

函数 VowelWord(),对 Word 中的 Na 个单词逐个测试,当包含元音时则用 Nb 计数。

```
VowelWord( )
  { char tag[5] ;
    int i,j,k,flag;
    for( i=1; i<=Na; i++ )
      { flag= 0;
        for( j=0; j<M; j++ )
          { strcpy( tag,"aeiou" ) ;
            for( k=0; tag[k] ! = '\0'; k++ )
              if( Word[i][j] = = tag[k] ){ flag=1;
                                           goto next;
                                         }
          }
        next: if( flag )Nb++ ;
      }
    printf("带元音的单词为 %d 个\n", Nb);
  }
```

函数 FourVowel(),对 Word 中的 Na 个单词逐个检查,仅当包含四个不同元音时 Nc 计数。

```
FourVowel( )
  { char tag[5] ;
    int i,j,count ;
    for( i=1; i<=Na; i++ )
      {
        strcpy( tag,"aeiou" ); count= 0;
        for( j=0; Word[i][j] ! = '\0'; j++ )
          if( FindDelete( tag,Word[i][j] ) ) count++ ;
        if(count= =4) Nc++ ;
      }
    printf("带四个不同元音的单词是 %d 个\n", Nc );
  }
```

函数 FindDelete()的执行,若当前 Word[i][j]中是元音,则将它从 tag 中清除,并返回删除标志值 flag。

```
FindDelete( char tag[], char ch )
  { int i,flag ;
    flag= 0;
    for( i=0; tag[i] ! = '\0'; i++ )
    if( tag[i] = = ch ){ tag[i]= ' ';
                         flag= 1;
```

```
    }
    return( flag );
}
```

用主函数 main()组织调用上述模块,调用策略为:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define  N      50
#define  M      15
char Word[N][M];
int Na,Nb,Nc;
DispWord( )
  { int i;
    for( i=1; i<=Na; i++ )
      printf("%18s\n", Word[i] );
  }
EnterText( char *buf )
  {
    gets( buf );
  }
main( )
  { char buf[80];
    int i,option,pos=0;
    do{
       printf("1__读待测试的文本  2__读字符拼单词  3__查看所拼单词\n");
       printf("4__检索带元音单词  5__带四个不同元音的单词  6__结束\n");
       scanf("%d", &option );
       switch( option )
         { case 1: EnterText( buf );
                   break;
           case 2: Na= 0;
                   while( buf[pos] != '\0' )
                     { ScanPos( buf,&pos );
                       if( buf[pos]=='\0' )break;
                       GetWord( buf,&pos );
                     }break;
           case 3: DispWord( );
                   break;
           case 4: Nb= 0;
                   VowelWord( );
                   break;
           case 5: Nc= 0;
                   FourVowel( );
                   break;
         }
```

```
    }while( option ! = 6 );
}
```

作为结构化程序设计方法的编程练习，请读者模仿例 1.1 的做法，设计一个程序，验证哥德巴赫猜想。哥德巴赫猜想的命题是：任一个大于等于 4 的偶数，都可以表示成两个素数之和。例如，4 = 2 + 2，8 = 3 + 5。

1.1.2 结构化程序的基本结构

一个结构化程序，总是由一些基本控制结构，按照一定的逻辑规则而有序地组织在一起的。这些基本结构为：顺序结构、选择结构、循环结构，并且实现单入口单出口。结构化程序设计，使程序的“控制流程线性化”，就是说程序动态执行的顺序符合静态书写结构，增强了程序可读性，方便调试和出错排除。

同样，表述问题的“计算逻辑”，即设计算法时，也用这三种基本结构。详见第 1.2 节讨论的算法。

1. 顺序结构

顺序结构的特点是：计算机执行这种程序，按书写的先后次序，自上而下地逐条执行，中间没有条件判定分叉和重复过程。见示意图 1.1。

2. 选择结构

选择结构的特点是：程序执行的控制出现了“分叉”，要根据不同情况选择其中之一执行，即“具体问题具体分析”。就是说，应该设计选择结构程序。见示意图 1.2。

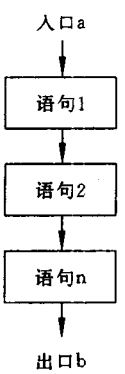
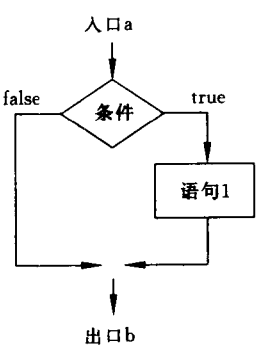


图 1.1 顺序结构



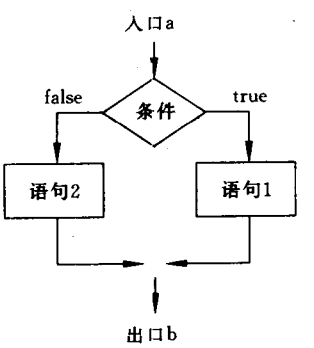


图 1.2 选择结构

3. 循环结构

较复杂问题的求解，往往要做大量的重复计算，即设计循环结构程序。循环结构由循环条件和循环体组成。在循环执行过程中，循环体中的语句没变，参加运算的数据变了，循环体在有限次运行后终止。见示意图 1.3。

在对实际问题做结构化程序设计时，上述三种基本结构常常是“联合”使用的，请看下面例子。

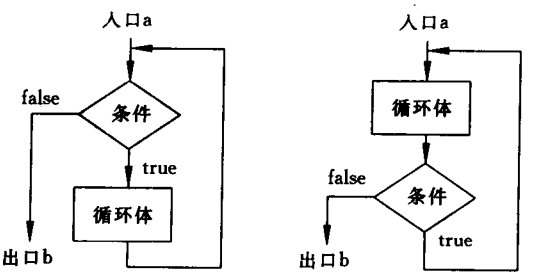


图 1.3　循环结构

【例 1.2】已知 s1,s2,...,sn 是一组成绩数列,将它们从低分到高分排序。假设成绩数据已存入数组 s,采用选择法作数据排序,算法程序为:

```
DataSort( float s[ ], int n )
  { int i,j,k;
    float temp;
    for( i=0; i<n-1; i++ )
      {
       k= i;
       for( j=i+1; j<n; j++ )
         if( s[k] > s[j] ) k=j;
       temp= s[i];
       s[i]= s[k];
       s[k]= temp;
      }
  }
```

1.1.3　程序的总体设计

一个较大问题的程序设计,在明确它"做什么"(要完成什么功能)以后,先做总体设计阶段工作,再做详细设计阶段工作。

总体设计是确定系统的整体模块结构和模块之间的逻辑关系。总体设计的任务是把系统功能需求分配给模块结构,总体设计的表达工具是模块层次结构图(如图 1.4),总体设计的基本方法仍然是"自顶向下,逐步求精"。

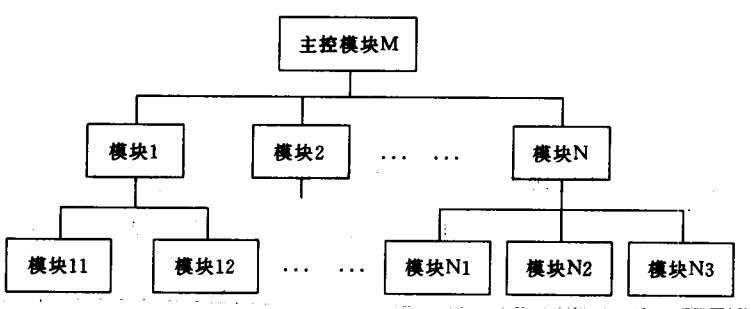


图 1.4　模块结构图

在模块结构图中，矩形框表示功能模块，上下层连线表示模块之间调用关系，按惯例是上层调用下层。

【例 1.3】设计学生成绩管理系统，用于存储管理各种类别（专科、本科、研究生）学生的成绩信息。

教务员管理学生成绩时，经常处理两张基本表格：单科成绩表、学生成绩总表。请见表 1.1 和表 1.2。

表 1.1　单科成绩表

专业：	年级：	类别：	课程名：	学年度：	
学　号	姓　名	成　绩	学　号	姓　名	成　绩

教师签字：

表 1.2　学生成绩总表

专业：	年级：	类别：	学号：	姓名：
学　年　度	课　程　名	成　绩	课　程　名	成　绩
第一学年：				
第二学年：				
第三学年：				
第四学年：				

系别：

单科成绩表记录了某学年、某学期、某门课的学生成绩，任课教师签字后交教务员。学生成绩总表汇总了任一个有学籍的学生在校期间的学习成绩，如专科生的三年学习成绩、本科生的四年学习成绩、研究生的三年学习成绩等，由系主任签字。

因此，学生成绩管理系统的数据结构可定义为：

```
#define Nr   5
#define Mr   4
#define Kr   9
struct Student { char spec[10];
```

```
    char grade[10];
    char style[10];
    char number[10];
    char name[10];
    struct Score { char year[11];
                   char km[Kr][11];
                   char cj[Kr][4];
                 }C[Mr];
    char total[10];
    char aver[10];
    char dept[10];
  } S[ Nr ] ;
char *Nian[4]={"第一学年","第二学年","第三学年","第四学年"} ;
```

其中,系别 dept,专业 spec,年级 grade,类别 style,学号 number,姓名 name,学年度 year,课程名 km,成绩 cj,total 总成绩,aver 总平均分。

结构数组 S 存放 Nr 个注册学生的数据信息,结构数组 C 登记 Mr 个学年中该生所学的 Kr 门课程的课程名和成绩。

用计算机协助教务员做学生成绩管理,应进行成绩的录入、汇总、分类统计,以及常规的的数据显示、排序、查询、删除或插入等操作。系统的总体设计结构框架见图 1.5。

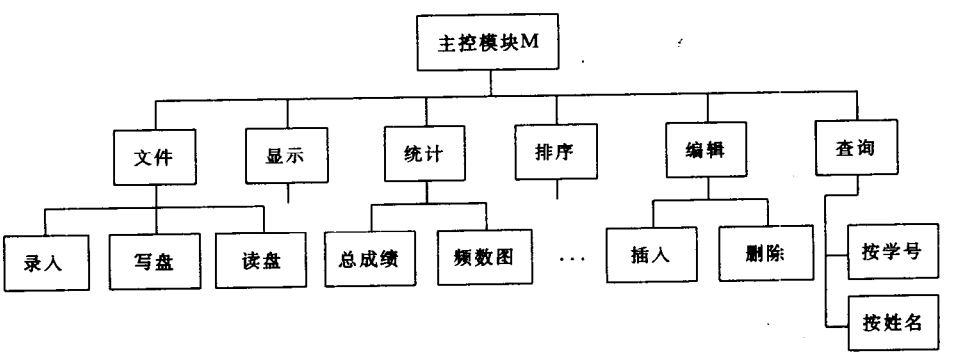


图 1.5 学生成绩管理模块结构示意图

1.1.4 程序的详细设计

总体设计所获得的模块结构仍是个“黑盒子”,也就是说知道了做什么,尚不知道问题实现的细节。

详细设计的任务是完成怎么做,给出模块的过程描述,即模块内部算法设计,用高级语言表达算法。详细设计的表达工具是用伪码书写模块的功能算法。伪码是一种介于某种高级语言和自然语言之间的控制结构,见下面第 1.2.2 节的叙述。

【例 1.4】学生成绩管理系统中的几个功能模块的算法描述。数据说明为例 1.3。

学生成绩管理系统的主控模块算法描述:

假设系统相关功能模块为:数据初始化 InitData(),信息显示 DispData(),信息查询 SearchData(),成绩统计 StatData(),记录删除 DelData(),记录插入 InsData(),按总分排序

SortData()。

```
main( )
  { int option;
    do{
       printf("_____主菜单_____\n");
       printf("1 ___数据初始化     2 ___信息显示\n");
       printf("3 ___信息查询       4 ___成绩统计\n");
       printf("5 ___记录删除       6 ___记录插入\n");
       printf("7 ___按总分排序     8 ___程序结束\n");
       scanf("%d",&option);
       switch(option)
         { case 1: 调用函数 InitData();
                   break;
           case 2: 调用函数 DispData( );
                   break;
           case 3: 调用函数 SearchData( );
                   break;
           case 4: 调用函数 StatData( );
                   break;
           case 5: 调用函数 DelData( );
                   break;
           case 6: 调用函数 InsData( );
                   break;
           case 7: 调用函数 SortData( );
                   break;
           case 8: printf("结束\n");
         }
      }while( option ! = 8 );
  }
```

学生成绩数据初始化的算法描述：

```
InitData( struct Student S[ ] )
  { int i,j,k ;
    for( i = 0; i < Nr; i++ )
      { 键入第 S[i]个学生的专业,年级,类别,学号和姓名数据;
        for( j = 0; j < Mr; j++ )
          { 键入第 S[i]个学生的第 C[j]个学年号;
            for( k = 0; k < Kr; k++ )
              键入该生的第 C[j]个学年所学的第 km[k]门课程及成绩 cj[k] ;
          }
      }
  }
```

统计每个学生的总成绩和平均分的算法描述：

```
StatData( struct Student S[ ] )
  { int i,j,k ;
```

```
    for( i=0; i<Nr; i++ )
      { 置第 S[i]个学生的总成绩单元 total 为空;
        for( j=0; j<Mr; j++ )
          for( k=0; j<Kr; k++ )
            将该生的第 C[j]个学年所学的第 km[k]门课程成绩 cj[k]累加到 total 中;
        将平均分 total/(Mr*Kr)存入 aver 内;
      }
  }
```

依此类推,可以方便地做其他几个模块的算法设计。学生成绩管理系统的诸模块的具体实施,详见第9章。

1.2 算法

用计算机解题涉及两大类算法。第一种是数值算法,例如,求定积分,解方程,矩阵运算等,这些经典算法,大部分用在科学计算上。第二种是非数值算法,例如,对一组数列进行排序,查找数列中的某个数,在数列中插入一个新数,软件工作的屏幕界面设计等,这些算法在管理科学中应用十分广泛,其中许多问题没有现成的求解公式,是人们关心的热点。本节将讨论算法的几个基本问题,并用结构化设计方法的三个基本结构表述算法。

1.2.1 算法和算法的特性

1. 算法

算法是一种计算逻辑,这种计算逻辑是对特定问题规定的运算序列。粗略而直观的讲,对某个特定的问题,提出求解它的公式和步骤,这就是算法。算法执行以后,得到问题的解,或者指出问题无解。

【例 1.5】求正整数 n 和 m 的最大公约数。

欧几里德给出了该问题的"面向计算机"的辗转相减算法:

(1) 输入 n、m 的值。

(2) 若 n > m,则 n = n - m。

(3) 若 m > n,则 m = m - n。

(4) 若 n = m,则算法结束,转第(5)步。否则,重复上述过程,转回第(2)步。

(5) 输出 n(或 m)中的值,它是所求的最大公约数。

用 C 语言表达上述算法的程序片段是:

```
scanf("%d%d", &n,&m);
while( n != m )
  {
    if( n>m ) n=n-m;
    if( m>n ) m=m-n;
  }
printf("result = %d\n", n );
```

2. 算法特性

算法应该是正确有效的,算法的表述应该是明确而无歧义性的,算法在有限次执行以后应该终止。

(1) 有穷性

包含有限的操作步骤,执行有限次操作之后终止。在例 1.5 中,操作步骤是五步,终止条件是 n = m。无穷次地执行操作,不是一个正确的算法。

(2) 确定性

每一步是确切定义的,欲执行的操作要清楚地规定,不能"模棱两可"。例如,若将例 1.5 中的第 2 步和第 3 步改成为:

若 n > m 则 n = n - m 否则 m = m - n

那就不一定对。算法的某一步运算结果不确定,不可能产生正确的计算结果。

(3) 有效性

算法中要实现的运算都是相当基本的、能精确地执行,即每一步应当有效地执行,并且产生有效和确定的结果。例如,16 位(bit)微机中,两个实型量(如,x = 0.0000000123, y = 0.0000000321)作相等比较时,计算机则视为 true,事实上 $x \neq y$。用 $|x - y| < \varepsilon$ 较合适。

(4) 零个或多个输入

输入是对算法要加工的对象赋初值。例 1.5 中对 n 和 m 输入数据。零个输入的情况也有,例如求 100 之内的素数时,可以这样写:

```
for( n=2; n< =100; n++ )
  if( prime(n) )输出素数 n;
```

此处,加工对象 n 是由 for(n = 2; n < = 100; n++)中的赋值表达式给定的。函数 prime() 的值为真,则 n 是素数。

(5) 一个或多个输出

输出是打印算法运算结果(称为算法的解),否则"一无所获"。

算法的特性,约束程序员去正确地书写算法,使之正确无误地执行,达到求解问题的预期效果。初学算法设计的读者,在书写实际问题求解算法时,往往"忘记"了算法特性,使得所写的算法不能得到"好的算法程序"实现,这是要特别注意的。

1.2.2 算法的描述

算法的描述,采用类程序设计语言来书写,它是介于自然语言和某种高级语言之间的控制结构,又称伪码。用伪码描述问题数据结构和程序控制结构时,可以运用自然语言的词汇,灵活自由地表示实际操作或条件判定,而不是立即"陷入"语言程序实现的细节,这就适应各种任务实施的需要,容易被广大程序爱好者所接受。

例如,用伪码表述求正整数 n 和 m 的最大公约数的欧几里德辗转相减算法是:

```
{
  给 n 和 m 赋初值;
  while( n≠m )
    {
```

```
        if( n>m )n=n-m;
        if( m>n )m=m-n;
      }
    输出结果 n 或 m 中的值;
}
```

用伪码表示的算法,比较容易转换成高级语言源程序。本书在书写算法时,将全部采用伪码表述,有时直接用汉语、英语和数学语言书写解题步骤。

1.2.3 算法的 C 语言表示

将伪码书写的算法"转换"为 C 高级语言的源程序,这就是实际问题的最终程序设计。例如,求正整数 n 和 m 的最大公约数的欧几里德辗转相减算法的 C 语言程序为:

```
gcd( int n,int m )
  { while(n ! = m)
      {
        if( n>m )n= n-m ;
        if( m>n )m= m-n ;
      }
      return( n );
    }
main( )
  {
    printf("最大公约数 = %d\n", gcd(6,15) );
  }
```

程序执行时输出结果为:

```
最大公约数 = 3
```

【例 1.6】设计生成"螺旋方阵"的算法程序。

例如 7×7 的螺旋方阵是图 1.6。图中以不同字体描绘了从左上角(1,1)开始,沿顺时针方向旋转的情况,共转了三圈又 1/4 格子,终止点(4,4)。

1	2	3	4	5	6	7
24	25	26	27	28	29	8
23	40	41	42	43	30	9
22	39	48	49	44	31	10
21	38	47	46	45	32	11
20	37	36	35	34	33	12
19	18	17	16	15	14	13

图 1.6 螺旋方阵示意图

解题思路:假设 N=7,矩阵 a 是 N×N 的方阵,i、j 分别表示动点的行位置和列位置,则动点转一圈途经四个方向,先后顺序为图 1.7。

若用 count 计数动点先后到达的格子序号,取值 1~49,当动点在(i,j)位置时,则矩阵元素

a[i,j] = count。按图 1.7 的走步规则，动点每转一圈，其行列位置 i 和 j 向“内”缩小一个值，直到 count = N * N 时终止。

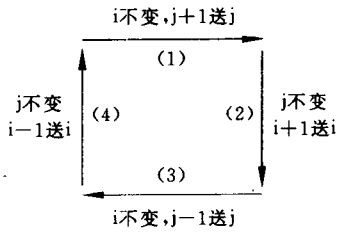


图 1.7　螺旋方阵求解示意图

算法描述：

(1) 从左向右移动

```
GotoRight( i,j,last,count )
  { int k;
    for( k= j; k< =last; k++ )
      { count++;
        a[i][k]= count;
      }
    last 是下一个方向的列位置 j ;
  }
```

(2) 自上向下移动

```
GotoBottom( i,j,last,count )
  { int k;
    for( k= i; k< =last; k++ )
      { count++;
        a[k][j]= count;
      }
    last 是下一个方向的行位置 i ;
  }
```

(3) 从右向左移动

```
GotoLeft( i,j,last,count )
  { int k;
    for( k= j; k> =last; k-- )
      { count++;
        a[i][k]= count;
      }
    last 是下一个方向的列位置 j ;
  }
```

(4) 自下向上移动

```
GotoTop( i,j,last,count )
  { int k;
    for( k= *i; k> =last; k-- )
      { count++;
        a[k][j]= count;
      }
    last 是下一个方向的行位置 i ;
  }
```

(5) 主控模块算法描述