高职高专计算机系列教材
Gaozhi Gaozhuan Jisuanji Xilie Jiaocai

C语言程序设计与数据结构实践

闵光太　主编

高等教育出版社
HIGHER EDUCATION PRESS

高等职业与高等专科教育计算机系列教材

C 语言程序设计与数据结构实践

闵光太　主编

张燕　王春红　武建华　马佳琳　陈觉婷　编

高 等 教 育 出 版 社

图书在版编目（CIP）数据

C语言程序设计与数据结构实践 / 闵光太主编. —北京：高等教育出版社，2000
ISBN 7-04-007933-X

Ⅰ.C… Ⅱ.闵… Ⅲ.①C语言-程序设计②数据结构 Ⅳ.TP312

中国版本图书馆CIP数据核字（2000）第06902号

C语言程序设计与数据结构实践
闵光太 主编

出版发行 高等教育出版社
社 址 北京市东城区沙滩后街55号
电 话 010—64054588
网 址 http://www.hep.edu.cn
邮政编码 100009
传 真 010—64014048

经 销 新华书店北京发行所
印 刷 北京二二〇七工厂

开 本 787×1092 1/16
印 张 22.25
字 数 546 000

版 次 2000年5月第1版
印 次 2000年5月第1次印刷
定 价 25.00元

前　言

为适应国家大力发展高等职业教育的形势，在教育部高教司的指导下，全国高等职业教育研究会教材编写小组和高等教育出版社联合组织编写出版高职系列教材。本教材是计算机应用专业的系列教材之一，由金陵职业大学主持编写。

高等职业教育是以培养技术应用型人才为目标的高等教育。在理论“必须、够用”的前提下着重培养学生的实际应用能力。按照这一基本要求，本教材旨在加强C语言程序设计和数据结构两门课的衔接，探索用数据结构算法的实现作为C语言程序设计应用的方法，把“算法”和“程序”紧密结合在一起，加强学生对实际问题抽象描述的理解能力，以及运用C语言解决实际问题的能力。本教材的各章基本上都从案例出发提出问题，引出本章的教学重点，围绕问题的解决展开本章的论述，并突出实践性教学环节，在每章最后一节设置实验单元。此外，在最后一章着重讲述了C语言实用技术，以使学生能将从本书所学的知识全面、综合地加以运用，有效地提高学生的程序设计技能。

本教材的主要内容是：第一章讲解结构化程序设计的思想、算法的概念、数据结构的基础知识和软件开发的基本过程；第二章为C语言概述；第三章为结构控制语句；第四章为数据的顺序存储结构及应用；第五章为函数；第六章为指针；第七章为数据的链式存储及应用；第八章为树的存储结构及应用；第九章为查找与排序算法；第十章为位运算；第十一章为文件；第十二章为实际工程设计中两个典型实例的分析。

本书根据作者多年教学讲稿和教学经验写成，其中的实例都在PC 386机器上用Turbo C 2.0调试通过。

参与本教材编写的作者有：第一、十、十一、十二章由南京金陵职业大学张燕老师编写，第二、三章由山西运城高等专科学校王春红老师编写，第四、五章由河南新乡平原大学武建华老师编写，第六、七章由辽宁沈阳大学马佳琳老师编写，第八、九章由厦门鹭江大学陈觉婷老师编写，全书由张燕负责统稿。

建议本教材的教学时数为96学时，其中实践时数48学时。

本教材由南京东南大学计算机科学与工程系王茜教授审订，王茜老师对本书提出了许多宝贵的意见和建议，在此谨向她以及关心本书编写的同志们表示衷心感谢。

本教材的编写是对计算机软件课程教学改革的一种探索，难免有不到之处，敬请各位专家和广大读者给予批评指正。

闵光太　　于金陵职业大学

1999.10.12

第一章 概 论

C 语言是近年来在国内外流行的高级程序设计语言，它既具有高级语言的特点，又具备低级语言的特点，它不仅被计算机软件工作者用来设计开发系统软件，而且也为广大的工程技术人员所喜爱。C 语言目前已在系统软件开发、科学工程数值计算、数据处理、计算机图形技术等方面得到了广泛应用。本章首先介绍结构化程序设计的基本方法；其次介绍数据结构与算法在程序设计中的地位和重要性；最后介绍 C 语言的特点、基本结构及调试方法。

1.1 结构化程序设计的基本方法

结构化程序设计是系统详细设计时使用的有效设计技术。它的任务是将经过改进的模块结构中的各个模块，用结构化设计方法设计出正确、合理的算法，并用一定的图示工具描述出来，为进一步编程作好准备。

1.1.1 结构化程序设计思想

结构化程序设计方法是 Dijkstra 于 1965 年提出来的。它的基本思想是要求程序设计者不要随心所欲地编写程序，而要按一定的结构形式来设计和编写程序，使程序易读、易理解、易修改。同系统分析和系统的概要设计时采用的分析方法一样，结构化程序设计方法也是采用自顶向下逐步求精的设计方法，把一个模块逐步分解细化为一系列的处理步骤，使之成为单入口和单出口的结构形式。

1966 年 Bobra 和 Jacopini 提出了程序的三种基本结构：顺序结构、分支结构和循环结构。这三种基本结构有以下特点：

1．只有一个入口。

2．只有一个出口。

3．结构内的每一部分都有机会执行到。也就是说，对每一个框来说，都应当有一条从入口到出口的路径通过它。

4．结构内无死循环。

已经证明，由三种基本结构组成的算法可以解决任何复杂的问题。由基本结构所构成的算法称为结构化的算法。采用结构化的算法进行程序设计实际上就是结构化的程序设计。

1.1.2 结构化程序设计的步骤

学习计算机语言的目的是利用该语言工具设计出可供计算机运行的程序。

注：鉴于计算机程序的习惯和特点——所有的文字和符号都用正体。因此，本书中所有的与计算机程序有关的文字和符号都采用正体或保持与软件开发工具中显示的一致——编辑注。

在拿到一个需要求解的实际问题之后，怎样才能编写出程序呢？以数值计算为例，一般按照图 1.1.1 所示的步骤进行。

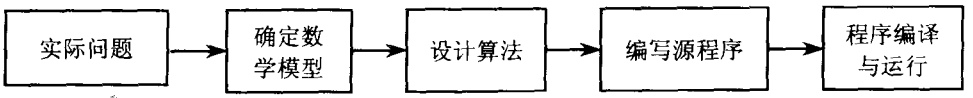


图 1.1.1

一般来说，从实际问题抽象出数学模型是有关领域专业工作者的任务。程序设计人员的工作，最关键的一步就是设计算法。一般以流程图来表示算法。如果算法是正确的，将它转换为任何一种高级语言程序是不困难的，这一步常称为“编码”。

结构化程序设计的步骤是：

1．分析问题

准确而完整地描述和分析问题是解决问题的第一步。该问题要求你设计的程序的整体功能是什么？涉及到哪些数据对象？

2．数据对象的描述

采用何种数据结构？用高级语言实现时，采用何种数据类型？

3．算法的设计

在设计结构化的算法时一般采用下述方法：

(1) 自顶向下

这种方法是先有全局，先进行整体设计，然后再进行下一层的设计，逐步地实现精细化。采用这种方法能做到胸有全局，通盘考虑，不致顾此失彼、头重脚轻。

(2) 逐步细化

一步步地将上层的任务分解成较小的、易于实现的任务，直到可以很简单地实现为止。

(3) 模块化

将一个大任务分解成若干个较小的部分，每一部分承担一定的功能，称为“功能模块”。各个模块都可以被独立地编写和调试。便于组织人力完成较复杂的任务。

4．编写源程序

这一步也称为“编码”，就是将已设计好的算法用高级语言源程序表达出来。

5．程序的编译与运行

编译的一项重要的功能就是检查源程序的语法错误，编译成功之后得到与源程序对应的目标程序。运行就是指运行可执行的目标文件。

下面举一个简单的例子来说明如何实现结构化程序设计。

例 1.1　给出 100 个连续的整数（每个数都≥3），打印出其中的素数。

问题分析及数据对象的描述　由于题目中问题明确，数据对象就是大于且等于 3 的 100 个整数，这 100 个整数可以以 100 个整型变量的形式进行处理，也可以以含 100 个分量的整型数组的形式来处理，哪种形式的数据处理起来简单高效就应选取相应的数据类型。

算法设计　采用自顶向下、逐步求精的方法来设计算法。先把问题分为三部分：① 输入 100 个数给 x_1 到 x_{100}；② 把其中的素数找出来（或者把非素数除去）；③ 打印出全部的素数，见图 1.1.2。这三部分分别用 A、B、C 表示。可以把这三部分分别以三个功能模块来实现。

图 1.1.2 所示的三部分内容还是笼统的、抽象的，因为还没有解决怎样才能实现“把素数找出来”的要求。需要进一步“细化”。对第一部分的细化可用图 1.1.3 表示。用 x_i 代表 x_1 到 x_{100} 中的某一个。i 的值由 1 变到 100。对第二部分的细化见图 1.1.4。对第三部分的细化见图 1.1.5。到此为止每一部分都可用高级语言中的语句来表示，整个程序流程图见图 1.1.6。

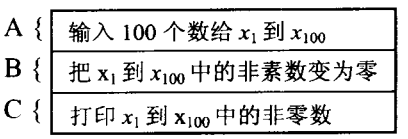


图 1.1.2

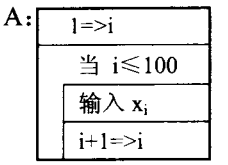


图 1.1.3

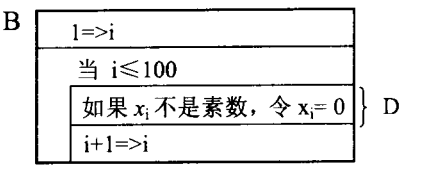
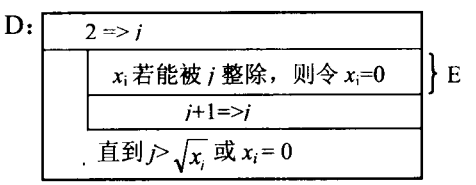
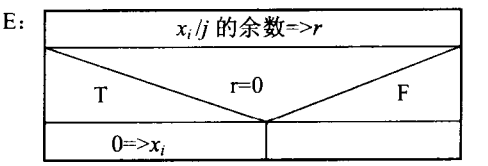
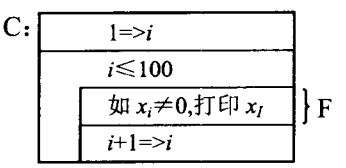


图 1.1.4







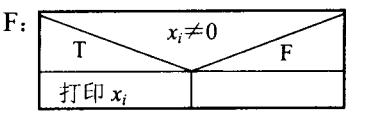


图 1.1.5

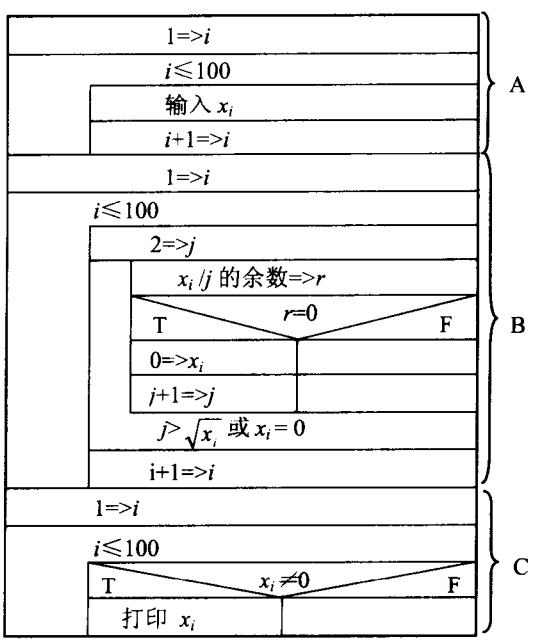


图 1.1.6

从图 1.1.2 到图 1.1.6 可以看出：逐步细化的过程一直继续到得到三种基本结构之一为止，因为基本结构不能再分解了。逐步细化是结构化程序设计方法的核心。应当学会利用这种方法把一个总的任务（即抽象的要求）逐步具体化。可以说，逐步细化的过程是使求解问题“由抽象到具体”的过程。

编写源程序　是一个由算法到高级语言程序的转变过程。上例算法所对应的 C 语言源程序 f.c 如下：

```
#include <stdio.h>
#include <math.h>
#define N 100
main()
{
    int i, j, line = 0, a[N], k = 0;
    for(i = 3; i <= N+2; i++)   a[i-2] = i;
    for(i = 1; i <= N; i++)
    {
        k = sqrt(a[i]);
        for(j = 2; j <= k; j++)
            if (a[i]%j == 0)
            {
                a[i] = 0;
                break;
            }
    }
    printf ("\n");
    for (i = 1; i <= N; i++)
    {
        if (a[i] != 0)
        {
            printf ("%5d", a[i]);
            line++;
        }
        if (line % 10 == 0 && a[i] != 0) printf("\n");
    }
}
```

程序的编译与执行　如果在编译过程中发现源程序有语法错误，系统会输出“出错信息”，告诉用户第几行有什么样的错误。用户应重新调用编辑程序进行修改后再进行编译。如此直到编译通过为止。执行就是执行已编译好的可执行文件。此时，屏幕上会显示出程序应输出的结果。如果程序需要输入数据，则应在此时输入数据，然后接着执行程序，输出结果。如果发现运行结果不对，要重新修改源程序，进行编辑、编译之后再执行。

1.2　数据结构与算法

1.2.1　数据结构

1．什么是数据

首先了解一下什么是数据。**数据**是现实世界中的各种信息记录下来的、可以识别的符号。它们是信息的载体，是信息的具体表示形式。数据可以是数字、文字、图像或其他特殊的符号，目的是要对它进行通信、解释和处理。数据是计算机程序使用和加工的“原料”。数据的基本单位是**数据元素**，在计算机程序中通常作为一个整体进行考虑和处理。它可以是一个单独的符号，例如一个英文字母；也可以由若干个**数据项**组成，如一个班级的全体学生学籍登记表（表 1.2.1），它的数据元素由学号、姓名、性别、年龄、职务五个数据项组成。数据项是数据不可分割的最小单位。

表 1.2.1　学籍登记表

学号	姓名	性别	年龄	职务
98321	王威	男	18	学习委员
98322	洪流	男	18	生活委员
……	……	……	……	……

性质相同的数据元素的集合叫做**数据对象**。数据对象中的元素彼此之间存在的相互关系叫做**结构**。

2．什么是数据结构

为了进一步理解什么是数据结构，我们来看一个具体的例子。图书馆是大家所熟悉的，那么图书馆的结构是什么样的呢？从物理上看，图书馆主要是由装书的书架和书籍组成，但另一方面，还存在一个图书馆的编目表，即图书馆藏书的索引。图书馆从两个方面管理图书：物理的藏书和逻辑的编目表——这就是图书馆的结构。和图书馆一样，计算机在管理数据时，也有两个方面：即物理的存储和逻辑的关系。从这两方面，我们来回答什么是数据结构。

数据结构是相互之间存在一种或多种特定关系的数据元素的集合。具体来说，它包括数据的物理结构和逻辑结构。

数据的**物理结构**：指数据元素在计算机存储器中的表示，即存储结构。比如向量、链表。

数据的**逻辑结构**：仅考虑数据元素间的逻辑关系。它包括线性结构（如线性表、栈、队列）和非线性结构（如树、二叉树和图）。

一种逻辑结构通过映像便得到它相应的存储结构。同一种逻辑结构可以映像成不同的内部存储结构。反过来，数据的存储结构一定要反映数据之间的逻辑关系。

关系（Relation）是指数据元素间的逻辑关系。

根据数据元素之间关系的不同特性，通常有下列四类基本结构：

(1) 集合　数据元素之间除了“同属于一个集合”的关系外，别无其他关系。

(2) 线性结构　对数据元素而言，只有一个前趋和一个后继（首末两个元素例外，第一

个元素无前趋，最后一个元素无后继）。

(3) 非线性结构　对数据元素而言，或者有一个前趋，多个后继（树形结构）；或者有多个前趋，多个后继（图或网状结构）。

数据元素之间的关系在计算机中有两种不同的表示方法：顺序映像和非顺序映像，并由此得到两种不同的存储结构：顺序存储结构和链式存储结构。顺序映像的特点是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系；非顺序映像的特点是借助指示元素存储地址的指针（Pointer）表示数据元素之间的逻辑关系。

数据的逻辑结构和物理结构是密切相关的两个方面，任何一个算法的设计取决于选定的数据逻辑结构，而算法的实现取决于采用的存储结构。

数据类型（Data Type）是和数据结构密切相关的一个概念，它最早出现在高级程序设计语言中，用以刻画（程序）操作对象的特性。在用高级语言编写的程序中，每个变量、常量、表达式都有一个它所属的特定的数据类型。类型明显或隐含地规定了在程序执行期间变量或表达式所有可能取值的范围，以及在这些值上允许进行的操作。因此，数据类型是一个值的集合和定义在这个值集上的一组操作的总称。

按“值”的不同特性，高级语言中的数据类型可分为两类：一类是非结构的原子类型。原子类型的值是不可分解的。如，C语言中的基本数据类型（整型、实型、字符型和枚举型）、指针类型和空类型。另一类是结构类型。结构类型的值是由若干成分按某种结构组成的，因此是可以分解的，并且它的成分可以是非结构的，也可以是结构的。例如数组的值由若干个分量组成，每个分量可以取各种类型。在某种意义上，数据结构可以看成是“一组具有相同结构的值”，则结构类型可以看成由一种数据结构和定义在其上的一组操作组成。

1.2.2　算法

1．算法及其特性

(1) 算法

所谓算法，就是解决某一类问题的办法。确切地说，就是对某一类特定的问题，给出解决该问题的一系列（有穷的）操作，而每一操作都有其确定性的意义，并在有限时间内可以计算出结果。一个算法有多个输入量，它是问题给出的初始数据，经过算法的实现，它有一个或多个输出量，这就是算法对输入运算的结果，即问题的解答。

(2) 特性

算法具有如下特点：

① 确定性　算法的每一个规则、每一个操作步骤都有确切的含义。即无二义性。

② 可执行性　算法的每一个操作，对计算机来说都是可执行的。

③ 有穷性　算法必须在有限步骤后可以计算出结果。

④ 输入　一个算法有零个、一个或多个输入，这些输入取自于某个特定的对象的集合。

⑤ 输出　一个算法有一个或多个输出。这些输出是同输入有着某些特定关系的量。

(3) 算法的评价

给定一个计算机系统后，一个程序就有两个资源：时间和空间。它们的使用受到三个因素的影响：硬件配置、算法的设置、问题的规模。在这三个因素中，算法是一个积极因素，

起着很大的作用。因此，对算法除要求是正确的以外，还要求它是高效率的，即占用内存空间少，所需运行时间短。于是，对算法的评价从两个方面进行：执行算法所需的时间长短；执行算法所需的计算机内存容量（即空间）大小。

如何确切的评价算法呢？这就要采用算法分析方法。算法分析的标准是算法的时间复杂度和空间复杂度。

2．算法的描述

设计一个算法以后，要对它进行描述，以便交流和阅读。常用的描述算法的方法有：自然语言、流程图、结构化流程图、伪代码。

(1) 自然语言

自然语言就是人们日常使用的语言，可以是汉语、英语或其他文字。用自然语言描述算法通俗易懂，但是它的缺点较多：

① 比较繁琐　往往要用一段繁琐的文字才能说清楚所要进行的操作。

② 容易出现“歧义性”　自然语言往往要根据上下文才能正确判断出其含义，不太严格。

③ 用自然语言描述顺序执行的步骤好懂。但如果算法中包含判断和转移时，用自然语言就不那么直观清晰。

例 1.2　用自然语言描述实现“求两个正整数 m 和 n 的最大公约数”的算法如下：

① 以 n 除 m，求 m/n 的整数商和余数 r_1。

② 判断余数 r_1 是否为 0？若 $r_1=0$，则 n 就是最大公约数。如果 $r_1\neq0$，则还未找到最大公约数，要继续做下去，执行第③步。

③ 将 n 除以 r_1，求 n/r_1 的整数商和余数 r_2。

④ 判断 r_2 是否为 0？若 $r_2=0$，则除数 r_1 就是最大公约数。如果 $r_2\neq0$，则还未找到最大公约数，要继续做下去，执行第⑤步。

⑤ 将 r_1 作为被除数，r_2 作为除数，再求 r_1/r_2 的整数商和余数 r_3。

⑥ 同第④步一样进行判断，若成功，则找到，若不成功，则继续进行，直到某一次除法 r_{n-1}/r_n 的余数为零时，r_n 即为最大公约数。

通过上面的例题，我们可以对自然语言描述算法的特点有一个了解。比如，第④步在用自然语言描述时，就不那么直观清晰。

(2) 伪代码

伪代码（pseudo code）使用一种介于自然语言和计算机语言之间的文字和符号来描述算法，它不用图形，因此比较紧凑，比较好懂。

例 1.3　对“求 10 个数中的最大数”，可以用伪代码表示如下：

```
read a
0=>a
do until n>=9
    read b
    if a<b
        b=>a
    n+1=>n
end do
print a
```

可以看出，伪代码的写法比较灵活，不像高级语言那样有严格的语法规则，在构思算法时可以用伪代码随手写来。与流程图相比，伪代码易于修改。

(3) 流程图

流程图是用图形来表示算法。用一些几何图形的框来代表不同性质的操作。ANSI 规定的一些常用流程图符号（见图 1.2.1）已被大多数国家接受。

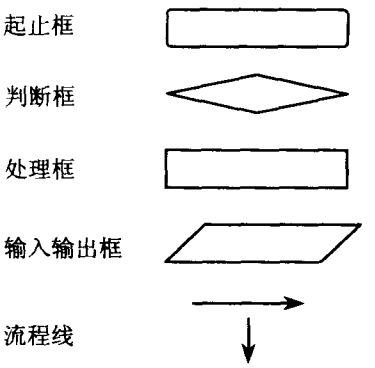


图 1.2.1　常用流程图符号

例 1.4　判断一个数是否是素数的算法用流程图表示如下。

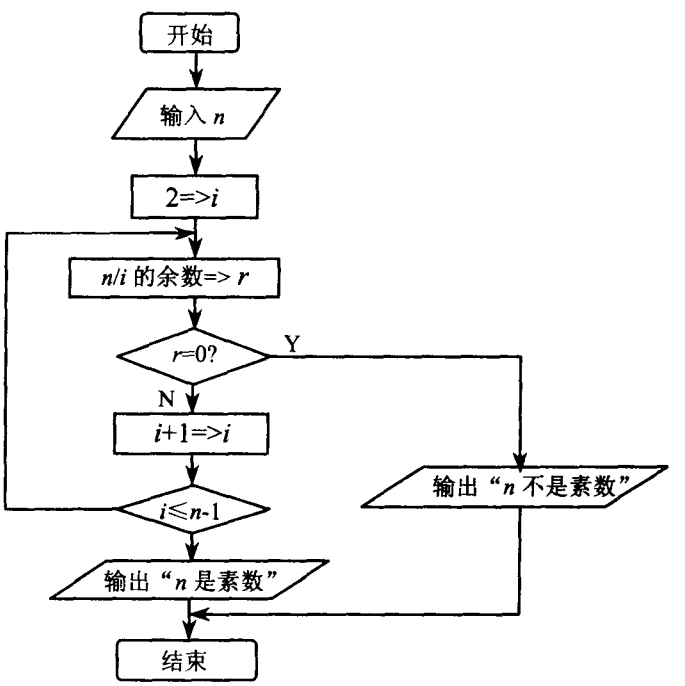


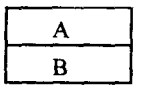
图 1.2.2

(4) N-S 结构流程图

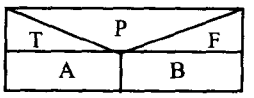
1973 年美国学者 I. Nassi 和 B. Shneiderman 提出了一种新的流程图形式。在这种流程图

中完全去掉了流程线，全部算法写在一个矩形框内，在框内还可以包含其他的框。这种流程图又称 N-S 结构流程图。这种流程图用三种基本元素框来表示三种基本结构。

① 顺序结构



② 选择结构



③ 循环结构

如图 1.2.3 (a)、(b)。

(a) (b)

图 1.2.3

例 1.5 将求最大公约数的算法用 N-S 结构流程图描述如下。

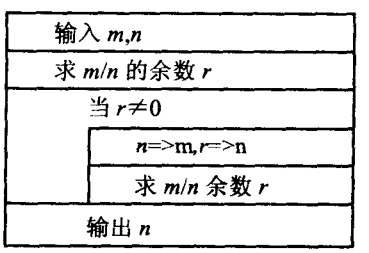


图 1.2.4

N-S 结构流程图具有明显的优点：

(1) 功能明确，即图中的每个矩形框所代表的特定作用域可以明确的分辨出来。

(2) 能够保证程序整体是结构化的，它不允许任意转移和设置出口，因此可以保证单入口、单出口的程序结构。

(3) 很容易实现和表示嵌套结构，这为较复杂的程序设计提供了方便的途径。

但由于 N-S 图仅使用三种基本结构形成程序，因此在某些程序设计时可能会比较繁琐，有一定的困难。

1.2.3 数据结构与算法

关于计算机科学，从 20 世纪 60 年代以来，一直有两种观点。Wegnor 代表一种观点，认为“计算机科学是一种基于信息结构转换的科学”。Knuth 代表另一种观点，认为“计算机科学是算法的学问”。这两种观点，从不同的侧面说明了“数据结构”、“算法”在计算机科学中的重要位置。实际上，数据结构和算法之间存在着本质的联系，应该说数据结构和算法是计算机科学的核心。

数据结构与算法之间的本质联系表现在两者互为依存。数据结构是为了研究数据运算而存在的；算法是为了实现数据运算，即实现数据的逻辑关系的变化，或是在这个结构上得到一个新的信息而存在的。进一步来讲，对数据结构而言，若不了解施加在数据上的算法，就无法决定实施算法的数据结构；对算法而言，若不了解基础的数据结构，就无法确定施加在数据结构上的操作（算法）。此外，数据结构与算法的本质联系还体现在提高计算机效率的作用上。数据结构直接影响计算机进行数据处理的效率。如前所述，算法的优劣也直接影响计算机的效率。

最后，数据结构与算法的本质联系是：算法 + 数据结构 = 程序。

实际上，一个程序除了以上两个要素外，还应当采用结构化程序设计方法进行程序设计，并且用某一种计算机语言来表示。因此上式可以改写为：

程序 = 算法 + 数据结构 + 程序设计方法 + 语言工具和环境

1.3 C 语言概述

本节对 C 语言的特点及 C 程序基本结构作简要的介绍。

1.3.1 C 语言的特点

1．C 语言的特点

C 语言是一种结构式、模块式、编译式通用程序设计语言，具有好的可移植性，适宜于编制各种软件（包括系统软件和应用软件），特别是系统软件。也就是说，各式各样的程序设计任务几乎都可以用 C 语言来完成。

Turbo C 是美国 Borland 国际有限公司于 1987 年首次推出的 C 语言，它具有 300 多个库子程序，其中包括分类子程序、目录子程序、进程子程序、转换子程序、诊断子程序、输入/输出子程序、接口子程序（DOS，8086，BIOS）、处理子程序（字符串，存储器）、数学子程序、存储分配子程序、杂子程序、标准子程序、时间和日期子程序等 12 类，以及 stdio.h、alloc.h 等 26 个头文件。在 C 程序中，可以用这些函数和包含文件来完成各式各样的程序设计任务，包括低级和高级的输入/输出、字符串和文件的处理、存储分配、进程控制、数据转换、数学计算等等。与传统的语言比，它的库函数丰富、启动快、使用容易、速度快（每分钟可编译 7 000 行）、效率高。欲建立和运行 C 语言程序，不需要使用单独的编辑程序、编译程序、连接程序以及 MAKE 软件，因为所有这些都已装进了 Turbo C 之内，构成了 Turbo C 的集成开发环境。Turbo C 可以在 IBM PC 包括 XT、AT 以及所有 IBM 兼容机上实现。Turbo C 支持美国国家标准协会（ANSI）提出的 C 标准，也完全支持克尼汉（Kernighan）和里奇（Ritchie）的定义，包括开发 PC 机所使用的混合语言和混合模式的程序设计的某些扩充。正因为 Turbo C 具有上述这些明显的优越性，所以它引起了人们的广泛重视。

由于 C 语言具有很多优于其他语言的特点，使得 C 语言具有强大的生命力，并且得到了不断的发展。其主要特点有以下几个方面：

(1) 语言简洁紧凑、使用方便灵活

C 语言一共只有 32 个关键字，9 种控制语句，程序书写形式自由，主要用小写字母，压

缩了一切不必要的成分。C 语言比其他高级语言设计的程序更简练，代码行少。语言的许多成分都通过函数调用完成，使得编译程序少而精。

(2) 丰富的运算能力

C 语言的运算符包含的范围很广泛，共有 34 种。C 把括号、赋值、强制类型转换等都作为运算符来处理。从而使 C 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) 丰富的数据类型

C 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的运算。因此，C 语言具有很强的数据处理能力。

(4) 结构化的语言

C 语言是一种结构化程序设计语言，即程序的逻辑结构可用顺序、分支、循环三种基本结构组成。其具有结构化的控制语句十分便于采用自顶向下、逐步求精的结构化程序设计方法。因此，用 C 语言编制的程序具有容易理解、便于维护等优点。

(5) 模块化的构造语言

C 语言程序的函数结构十分有利于把整个程序分割成若干个相对独立的功能模块，并且对程序间的相互调用、数据传递和数据共享方面提供了十分有效的手段。这一特点为把大型软件模块化、对由多人集体开发的软件工程提供了强有力的支持。

(6) 拥有完好的集成开发环境，交互式的编辑程序

编译、连接、运行能放在一起连续进行，而且启动快、编译速度快、易学易用。

(7) 具有低级语言的许多功能

C 语言允许直接访问物理地址，能进行位操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此 C 具有高级语言的特点，又具有低级语言的许多功能，可以用来写系统软件。

(8) 可移植性好

程序的可移植性是指在一定环境下运行的程序可以不加修改的在另一个完全不同的环境中运行。汇编语言依赖于机器的硬件，用汇编语言编写的程序不可移植。而一些高级语言，当它们从一种机器搬到另一种机器上时，基本上都要根据国际标准重新改写。而 C 语言程序本身并不依赖于机器硬件系统，从而便于在硬件结构不同的机器间和各种操作系统上实现程序的移植。

1.3.2　C 语言的程序结构

C 程序由一个或多个函数所组成，这些函数完成实际所需的操作。下面，我们通过一个简单的例子来具体说明 C 程序的基本结构。

例 1.6　输入数据 a、b 之后，试计算下式：x = (a*a+a*b) / (2*a–b)

程序如下：

```
/*program: 11.c*/                          /* 注释行 */
# include <stdio.h>                        /* 标准输入/输出头文件 */
main ( )                                   /* 主函数 */
```

```
{
    float a,b,x;                                    /* 变量说明 */
    print ("Enter two numbers: ");                  /* 输出字符串 */
    scanf ("%f%f", &a, &b);                         /* 输入变量的值 */
    x= (a*a+a*b)/(2*a-b);                           /* 算术运算并赋值 */
    printf ("a=%6.2fb=%6.2fx=%6.2f\n", a, b, x);    /* 输出结果 */
}
```

程序说明：

1．main（）表示“主函数”，每一个 C 语言程序必须有且仅有一个 main（）函数。

2．由大括号{}括起来的是函数体。

3．printf（）是 C 语言标准的输入输出库函数。

4．scanf（）是 C 语言标准的输入输出库函数。

5．/*……*/是 C 程序中的注释。

此程序的功能是输入变量 a 和 b 的值，然后输出算术表达式的值。

例 1.7　输入 a 和 b 两个数，求其最大者。

程序如下：

```
#include <stdio.h>
main ( )                                    /* 主函数 */
{
    int a, b, c;                            /* 变量说明 */
    scanf ("%d,%d", &a, &b);                /* 输入变量 a 和 b 的值 */
    c = max (a, b);                         /* 调用函数 max，将结果赋值给变量 c */
    printf ("max=%d\n", c);                 /* 输出 c 的值 */
}

int max (x, y)                              /* 定义 max 函数，函数值为整型，x，y 为形参 */
int x, y;                                   /* 对形参 x，y 作类型定义 */
{
    int z;                                  /* 对函数中用到的变量 z，加以定义 */
    if (x>y)   z = x;
    else   z=y;
    return (z);                             /* 将 z 的值返回，通过 max 带回调用处 */
}
```

程序说明：

1．本程序由两个函数构成：主函数 main（）和被调用函数 max（）。

2．max（）函数的作用是将 x，y 中较大的值赋给变量 *z*，返回语句 return 将 *z* 的值返回给主函数 main（）。返回值是通过函数名 max（）带回到主函数的调用处。

通过上述例子可以看出，C 语言程序具有以下结构特点：

(1) C 语言程序的基本单位是函数。一个 C 语言程序是由一个或多个函数构成的，其中必须包含一个主函数 main（）。主函数可以调用其他函数，被调用的函数可以是系统提供的库函数，也可以是用户自己定义的函数。

(2) 一个函数由两部分组成：

① 函数说明部分。包括函数名、函数类型、函数参数名、函数参数说明；

② 函数体。由一对大括号括起来的若干语句构成。通常这些语句分为两类：一类为变量定义，作用是定义函数中用到的变量；另一类为执行语句，作用是完成一定的算法处理。在某些情况下，可以既无变量定义也无执行语句，只有一对大括号构成一个空函数体，它与函数说明部分一起组成一个空函数，即什么也不做，这也是合法的。

③ 一个 C 语言程序总是从 main（）函数开始执行，这与 main（）函数在整个程序中的位置无关。main（）函数中所有的语句执行完成，则程序结束。

④ C 语言的每个语句、说明及变量定义之后都必须以分号结尾，分号是语句的必要组成部分。

⑤ 为便于理解和阅读 C 语言程序，可以用/*……*/（注释可以跨行）或//（注释必须在一行内完成）对 C 语言程序中的任何部分作注释。

1.3.3 程序的调试

程序的调试是一项复杂而又细致的工作，它要求程序调试者既要善于观察和思考，又要善于总结和积累经验。通过程序调试这一环节，可以进一步提高一个人的程序设计能力。因此，对初学者来说，切不可只注重设计出程序，而不重视程序调试这一重要环节。下面将简要介绍程序调试的典型步骤。

1．调试前的准备

(1) 熟悉程序运行环境

任何一种程序设计语言总是在一定的硬件和软件环境下进行编辑、编译、连接和运行的，而程序的调试和运行效率往往与对这些环境的熟悉程度有着直接的关系。因此，在调试运行程序之前，应该熟悉程序的编辑、编译、连接和运行的操作过程。例如，在程序调试过程中，可能要经常修改程序中的错误，而修改程序中的错误要修改编辑程序，如果对编辑程序的功能和基本操作不熟悉，必然要影响程序调试工作的顺利进行。又如，如果对程序的编译和连接过程（包括命令和操作）不熟悉，只作了编译而没有连接就想运行，或者个别模块还没有编译就想连接并运行等，都会产生错误。甚至由于对操作命令不熟悉，把一个已调试好的程序误删除，造成不必要的损失。所有这些问题虽然不是程序本身的错误，而是由于操作失误引起的，但也增加了程序调试的复杂性，影响了程序调试的效率。因此，在上机调试程序之前，必须认真仔细的了解和熟悉程序运行的环境，对于初学者还应练习一些常用的操作命令。

(2) 在程序设计过程中要为程序的调试作好准备

程序调试在整个程序设计过程中占有很大的比重，因此在程序设计一开始就应考虑到程序调试的一些措施，为程序调试创造必要的条件。主要体现在以下几个方面：

① 采用模块化、结构化设计方法设计程序。在设计中要按照不同的功能将程序模块化，尽量使一个模块完成单一的功能，并且要将结构化程序设计的原则贯穿于程序设计的始终。这样设计出来的程序便于调试。

② 编写程序时要为调试提供足够的灵活性。程序设计虽然是针对实际问题，但也要充