21世纪高等学校计算机规划教材

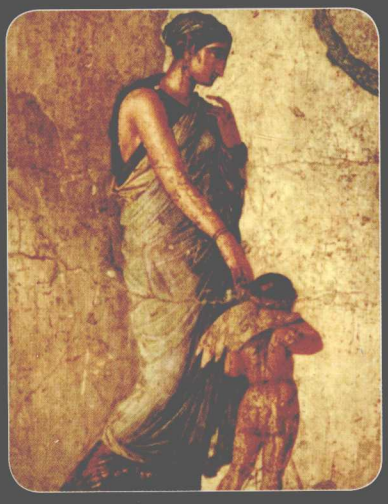
21st Century University Planned Textbooks of Computer Science

C语言程序设计教程（第3版）

The C Programming Language (3rd Edition)

徐士良 编著

- 由浅入深逐步介绍基本概念和语法
- 叙述简明扼要，文字通俗易懂
- 例题丰富实用，利于读者自学



名家系列

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

C语言程序设计教程（第3版）

The C Programming Language (3rd Edition)

徐士良 编著

名家系列

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

C语言程序设计教程 / 徐士良编著. —3版. —北京：人民邮电出版社，2009.10
21世纪高等学校计算机规划教材
ISBN 978-7-115-20005-1

Ⅰ. C… Ⅱ. 徐… 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（2009）第124854号

内 容 提 要

本书是作者通过长期教学实践而写成的。全书由浅入深，逐步介绍C语言中的基本概念和语法，使读者全面系统地理解和掌握用C语言进行程序设计的方法。主要内容包括：程序设计基本概念，C语言基本数据类型与基本输入输出，C语言表达式与宏定义，选择结构，循环结构，模块设计，数组，指针，结构体与联合体，文件，位运算。本次再版以Visual C++ 6.0环境为基础进行修订，内容更丰富，叙述更详细，更有利于读者自学。

本书叙述简明扼要，通俗易懂，例题丰富。本书可作为各专业的学生学习C语言程序设计的教材。

21世纪高等学校计算机规划教材

C语言程序设计教程（第3版）

◆ 编　　著　徐士良
责任编辑　刘　博

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市潮河印业有限公司印刷

◆ 开本：787×1092　1/16
印张：17.5
字数：454千字　　2009年10月第3版
印数：33 000－36 000册　　2009年10月河北第1次印刷

ISBN 978-7-115-20005-1/TP

定价：29.80元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

出版者的话

计算机科学与技术日新月异的发展，对我国高校计算机人才的培养提出了更高的要求。许多高校主动研究和调整学科内部结构、人才培养目标，提高学科水平和教学质量，精炼教学内容，拓宽专业基础，优化课程结构，改进教学方法，逐步形成了“基础课程精深，专业课程宽新”的良性格局。作为大学计算机教材建设的生力军，人民邮电出版社始终坚持服务高校教学、致力教育资源建设的出版理念，在总结前期教材建设的成功经验的同时，深入调研和分析课程体系，并充分结合我国高校计算机教育现状和改革成果，推出“推介名师好书，共享教育资源”的教材建设项目，出版了“21世纪高等学校计算机规划教材”名家系列。

本套教材的突出特点如下：

（1）作者权威 本套教材的作者均为国内计算机学科中的学术泰斗或高校教学一线的教学名师，他们有着深厚的科研功底和丰富的教学经验。可以说，这套教材汇聚了众师之精华，充分显示了这套教材的格调和品位。无论是刚入杏坛的年轻教师，还是象牙塔内的莘莘学子，细细品读其中的章节文字，定会受益匪浅。

（2）定位准确 本套教材是为普通高等院校的学生量身定做的精品教材。具体体现在：一是本套教材的作者长期从事一线科研和教学工作，对高校教学有着深刻而独到的见解；二是本套教材在选题策划阶段便多次召开调研会，对普通高校的教学需求和教材建设情况进行充分摸底，从而保证教材在内容组织和结构安排上更加贴近实际教学；三是组织有关作者到较为典型的普通高等院校讲授课程教学方法，深入了解教师的教学需求，充分把握学生的理解能力，以教材内容引导授课教师严格按照科学方法实施教学。

（3）教材内容与时俱进 本套教材在充分吸收国内外最新计算机教学理念和教育体系的同时，更加注重基础理论、基本知识和基本技能的培养，集思想性、科学性、启发性、先进性和适应性于一身。

（4）一纲多本，合理配套 根据不同的教学法，同一门课程可以有多本不同的教材，教材内容各具特色，实现教材系列资源配套。

总之，本套教材中的每一本精品教材都切实体现了各位教学名师的教学水平，充分折射出名师的教学思想，淋漓尽致地表达着名师的教学风格。我们相信，这套教材的出版发行一定能够启发年轻教师们真正领悟教学精髓，教会学生科学地掌握计算机专业的基本理论和知识，并通过实践深化对理论的理解，学以致用。

我们相信，这套教材的策划和出版，无论在形式上还是在内容上都能够显著地提高我国高校计算机专业教材的整体水平，为培养符合时代发展要求的具有较强国际竞争力的高素质创新型计算机人才，为我国普通高等教育的计算机教材建设工作做出新的贡献。欢迎各位老师和读者给我们的工作提出宝贵意见。

人民邮电出版社

第 3 版前言

本书第 1 版与修订版出版后，经过多年的使用，得到了各院校广大教师和读者的肯定。我们收到了许多读者的宝贵的意见，在此向他们表示感谢。

根据大多数教师和读者的意见，在修订版的基础上，第 3 版作了如下几个方面的改进。

（1）上机环境改为 Visual C++ 6.0。基本数据类型等内容作相应的修改和调整。

（2）增加了 C 基本数据类型在计算机中表示的内容。

（3）适当增加了一些例题。

（4）适当增加了一些习题。

总之，通过这次修订，使本书内容更丰富，叙述更详细，更有利于读者自学。

由于作者水平有限，书中难免还会存在错误和不妥之处，再次恳请读者批评指正。

编　者

2009 年 5 月

目 录

第1章 绪论

1.1 程序设计语言的发展

人们要利用计算机解决实际问题，无论是对于数值计算问题还是非数值计算问题，一般总是需要编制解决问题的程序。所谓程序，是指用某种程序设计语言为工具编制出来的动作序列，它表达了人们解决问题的思路，也反映了需要计算机所作的一系列操作。因此，程序设计语言实际上就是用户用来编写程序的语言，它是人与计算机之间交换信息的工具。

根据程序设计语言对问题的处理方式，程序设计语言一般分为机器语言、汇编语言和高级语言三大类。

1. 机器语言

对于计算机来说，一组机器指令就是程序，称为机器语言程序。

机器语言是最底层的计算机语言。用机器语言编写的程序，计算机硬件可以直接识别。在用机器语言编写的程序中，每一条机器指令都是二进制形式的指令代码。在指令代码中一般包括操作码和地址码，其中操作码告诉计算机作何种操作，地址码则指出被操作的对象。对于不同的计算机硬件（主要是 CPU），其指令系统是不同的，因此，针对一种计算机所编写的机器语言程序不能在另一种计算机上运行。由于机器语言程序是直接针对计算机硬件的，因此它的执行效率比较高，能充分发挥计算机的速度性能。但是，用机器语言编写程序的难度比较大，容易出错，而且程序的直观性比较差，也不容易移植。

2. 汇编语言

为了克服用机器语言编写程序的缺点，人们采用能帮助记忆的英文缩写符号（称为指令助记符）来代替机器语言指令代码中的操作码，用地址符号来代替地址码，以便于理解与记忆。用指令助记符及地址符号书写的指令称为汇编指令（也称符号指令），而用汇编指令编写的程序称为汇编语言源程序。汇编语言又称符号语言。

汇编语言与机器语言一般是一一对应的，因此，汇编语言也是与具体使用的计算机有关的。由于汇编语言采用了助记符，因此，它比机器语言直观，容易理解和记忆，用汇编语言编写的程序也比机器语言程序易读、易检查、易修改。

例如，为了计算表达式“3+2”的值，用汇编语言编写的程序与用机器语言（8086CPU 的指令系统）编写的程序如下：

```
PUSH  BP          01010101
MOVE  BP, SP      10001011  11101100
```

```
DEC  SP            01001100
DEC  SP            01001100
PUSH  SI           01010110
PUSH  DI           01010111
MOVE  DI, 0003     10111111  00000011  00000000
MOVE  SI, 0002     10111110  00000010  00000000
MOVE  AX, DI       10001011  11000111
ADD  AX, SI         00000011  11000110
MOVE  [BP-02], AX  10001001  01000110  11111110
POP  DI            01011111
POP  SI            01011110
MOVE  SP, BP       10001011  11100101
POP  BP            01011110
RET                11000011
```

其中每一行的前半部分为汇编语言指令，后半部分（二进制形式的指令代码）为对应的机器语言指令。

需要指出的是，计算机不能直接识别用汇编语言编写的程序，必须由一种专门的翻译程序将汇编语言源程序翻译成机器语言程序后，计算机才能识别并执行。这种翻译的过程称为“汇编”，负责翻译的程序称为汇编程序。

3. 高级语言

机器语言和汇编语言都是面向机器的语言，一般称为低级语言。低级语言对机器的依赖性太大，用它们开发的程序通用性很差，普通的计算机用户也很难胜任这一工作。

在保证程序正确的前提下，程序设计的主要目标是程序的可读性、易维护性和可移植性。为使程序的可读性好，不仅要求编程者具有良好的编程风格，还要求所使用的编程语言具有易懂的语句。所谓程序易维护，是指当程序的功能需要修改或增强时，所需要的开销应尽量的小。一个可移植性好的程序，应该在各种计算机和操作环境中都能运行，并得到同样的结果。显然，前面所提到的机器语言程序与汇编语言程序很难达到这样的要求。

随着计算机技术的发展以及计算机应用领域的不断扩大，计算机用户的队伍也在不断壮大。为了使广大的计算机用户也能胜任程序的开发工作，从 50 年代中期开始逐步发展了面向问题的程序设计语言，称为高级语言。高级语言与具体的计算机硬件无关，其表达方式接近于被描述的问题，易为人们接受和掌握。用高级语言编写程序要比低级语言容易得多，并大大简化了程序的编制和调试，使编程效率得到大幅度的提高。高级语言的显著特点是独立于具体的计算机硬件，通用性和可移植性好。

目前，计算机高级语言已有上百种之多，得到广泛应用的有十几种，并且，几乎每一种高级语言都有其最适用的领域。表 1.1 列出了几种最常用的高级语言及其最适用的领域。

表 1.1

语 言 名 称	适 用 范 围
BASIC	教学和小型应用程序的开发
FORTRAN	科学及工程计算程序的开发
PASCAL	专业教学和应用程序的开发
C	中、小型系统程序的开发
COBOL	商业与管理应用程序的开发
dBASE	数据库管理程序的开发

续表

语 言 名 称	适 用 范 围
FoxBASE	数据库管理程序的开发
C++	面向对象程序的开发
LISP	人工智能程序的开发
PROLOG	人工智能程序的开发
JAVA	面向对象程序的开发

为了计算表达式“3+2”的值，如果使用高级语言来编程就简单得多。

例如，用 BASIC 语言编写的程序为

```
10  I=3
20  J=2
30  K=I+J
```

又如，用 Pascal 语言编写的程序为

```
Program  Addit
var
   i, j, k: integer;
begin
    i: =3;
    j: =2;
    k: =i+j;
end
```

再如，用 C 语言编写的程序为

```
void main()
{ int  i, j, k;
  i=3;
  j=2;
  k=i+j;
}
```

由上述例子可以看出，程序设计语言越低级，就越靠近计算机硬件，其描述的程序就越复杂，其中的每一条指令（或语句）也就越难懂。反之，程序设计语言越高级，就越靠近人的表达与思维，其描述的程序就越简单，其中的每一条语句也就越容易理解。

必须指出，用任何一种高级语言编写的程序（称为源程序）都要通过编译程序翻译成机器语言程序（称为目标程序）后计算机才能执行，或者通过解释程序边解释边执行。

从程序设计语言的发展过程可以看出，程序设计语言将越来越接近人的自然语言。

1.2　程序设计的基本过程

什么是程序设计？对于初学计算机的人来说，往往简单地把它理解为编制一个程序。其实这是不对的，至少是不全面的。实际上，程序设计包括多方面的内容，而具体编制程序只是其中的一个方面。有人将程序设计描述成如下的一个公式：

程序设计=算法+数据结构+方法+工具

由此可以看出，在整个程序设计的过程中，要涉及算法的设计、数据结构的设计、方法的设计和设计工具的选择等诸多方面。从这个概念出发，一般来说，可以将程序设计的过程分为以下5个基本步骤：

1. 问题的分析；
2. 结构特性的设计；
3. 算法的设计；
4. 流程的描述；
5. 调试与运行。

下面分别对这5个步骤作简要的说明。

1. 问题分析

问题分析是进行程序设计的基础。如果在没有把所要解决的问题分析清楚之前就想着动手编制程序，是很难得到预想结果的，这只能起到事倍功半的效果。根据所要解决的问题性质与类型，需要分析的内容可能是不同的，但作为最基本的分析内容主要有以下几个方面。

（1）问题的性质

人们所要解决的问题是各种各样的，而对于不同性质的问题，所用的方法、工具以及输入输出的形式一般也是不同的。通过对问题性质的分析，进一步确定在解决这个问题过程中要做些什么？怎么做？例如，你所要解决的问题是属于数值型还是非数值型的问题；如果是数值型的问题，则要求确定一个合理的精度要求；不管是数值型问题还是非数值型问题，都需要明确最终的结果是什么。又如，对于一元二次方程求根的问题，需要明确是只求实根还是实根与复根都需要求。

（2）输入/输出数据

数据处理是计算机应用中最广泛的一个领域。在用计算机解决问题时，一般总要有一些输入数据，计算的结果也要以某种方式进行输出。因此，在进行程序设计的过程中，需要考虑对输入/输出数据的处理。一般来说，对于输入/输出数据主要应考虑以下几个方面：

- 数据的类型是什么？如整型、实型、双精度型、字符型等。
- 在何种设备上进行输入或输出？
- 采用什么样的格式进行数据的输入或输出？

（3）数学模型或常用的方法

对于数值型问题，一般要考虑数学模型的设计，或者要对常用的一些方法进行分析与比较，从而根据问题的性质选择一种合理的解决方案。对于非数值型问题，通常也需要从众多的方法中选择一种最合理的方法。

例如，为了求一元二次方程 $Ax^2+Bx+C=0$ 的两个实根 x_1 和 x_2，通常有以下3种方法：

① 求根公式

$$x_{1,2}=\left(-B\pm\sqrt{B^2-4AC}\right)\div(2A)$$

② 韦达定理

$$x_1+x_2=-B/A$$
$$x_1x_2=C/A$$

③ 迭代法

对于上述3种方法，虽然从理论上讲都可以使用，但与实际应用之间还是有一定的差距。例

如，因为计算工具的有效数字位数是有限的，在运算过程中不可避免地会出现误差，因此，利用理论上精确成立的求根公式计算得到的结果也不一定可靠；韦达定理虽然指出了一元二次方程两个实根之间的关系，但没有指明如何实现的具体步骤；迭代法一般一次只能求一个实根，并且还存在收敛性的问题。因此，在实际解决问题时，必须要对各种方法进行分析比较，不能随便使用某种方法求解。

2. 结构特性的设计

结构特性设计得好坏，直接影响到程序设计的效率，乃至程序执行的效率。结构特性的设计主要包括控制结构和数据结构的设计。

（1）控制结构

一个程序的功能不仅取决于所选用的操作，而且还取决于各操作之间的执行顺序，即程序的控制结构。程序的控制结构实际给出了程序的框架，决定了程序中各操作的执行顺序。在程序设计过程中，通常用流程图形象地表示程序的控制结构。常用的流程图有两种：传统流程图与 NS 结构化流程图（简称 NS 图）。

1966 年，Bohm 和 Jacopini 证明了任何复杂的程序都可以用顺序、选择和循环三种基本结构组合而成。其中选择结构中包括一般的选择结构和多情况选择结构，循环结构又可以分为当型循环和直到型循环两种。这几种基本控制结构的传统流程图如图 1.1 所示。其中：

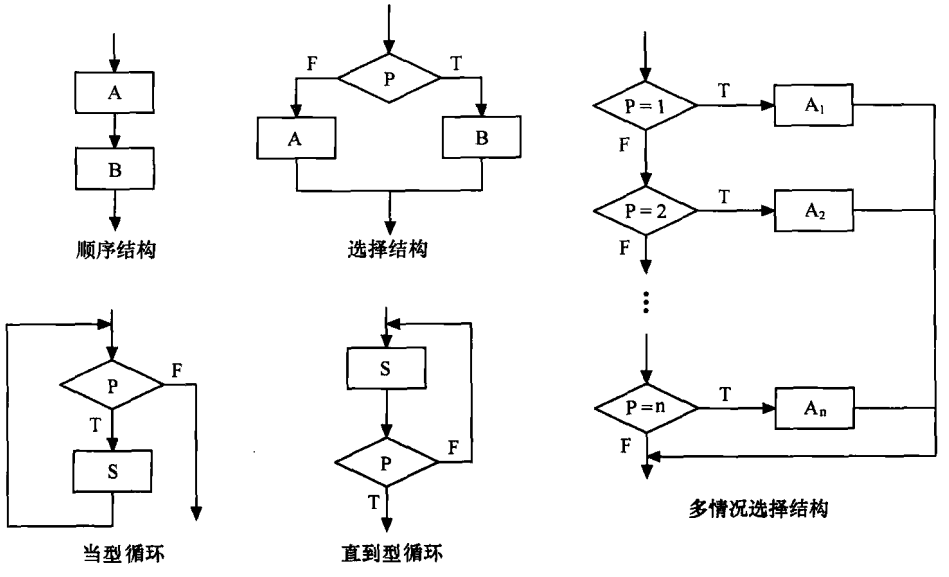


图 1.1　基本控制结构的传统流程图

- 顺序结构反映了若干个模块之间连续执行的顺序；
- 在选择结构中，由某个条件 P 的取值来决定执行两个模块之间的哪一个；
- 在当型循环结构中，只有当某个条件成立时才重复执行特定的模块（称为循环体）；
- 在直到型循环结构中，重复执行一个特定的模块，直到某个条件成立时才退出该模块的重复执行；
- 在多情况选择结构中，根据某控制变量的取值来决定选择多个模块中的哪一个。

传统流程图有以下几个主要缺点：

① 传统流程图本质上不是逐步求精的好工具，它会使程序员过早地考虑程序的控制流程，而

不去考虑程序的全局结构。

② 传统流程图不易表示层次结构。

③ 传统流程图不易表示数据结构和模块调用关系等重要信息。

④ 传统流程图中用箭头代表控制流，因此，程序员不受任何约束，可以完全不顾结构程序设计的思想，随意进行转移控制。

在 1.3 节中将介绍结构化流程图，即 NS 图。

（2）数据结构

在计算机的各种应用中，数据处理所占的比重将越来越大。在实际应用中，需要处理的数据元素一般有很多，而且，各数据元素之间不仅具有逻辑上的关系，还具有在计算机中实际存储位置上的关系。显然，杂乱无章的数据是不便于处理的，而将大量数据随意地存放在计算机中，实际上也是“自找苦吃”，对处理也是很不利的。

一般来说，在对数据进行处理时，数据的不同组织形式，其处理的效率是不同的。有关数据结构的问题有专门的课程介绍，在此不再详述了。

3. 算法的设计

前面已经提到，在进行问题分析时，要建立数学模型或对常用的方法进行分析比较，这实际上就是算法的设计。

所谓算法，是指解题方案的准确而完整的描述。从程序角度来看，也可以说算法是一个有限条指令的集合，这些指令确定了解决某一特定类型问题的运算序列。

对于初学程序设计的人来说，往往不加思考，随便拿来一种方法就用，最后导致结果不理想，甚至会得到错误的结果。

选择一个好的算法是程序设计的关键。选择算法主要应考虑以下两个基本原则：

（1）实现算法所花费的代价要尽量的小，即计算工作量要小。

（2）根据算法所得到的计算结果应可靠。

4. 流程的描述

程序设计的过程，实际上就是确定解决问题的详细步骤，而这些步骤通常称为流程。在程序设计过程中，常用的流程描述工具有以下几种。

（1）自然语言

自然语言是人们在日常生活、工作、学习中通用的语言，一般不需专门的学习和训练就能理解用这种语言所表达的意思。但是，用自然语言描述一个流程时，一般要求直接而简练，尽量减少语言上的修饰。

【例 1.1】 计算并输出 $z=y/x$。用自然语言描述其流程如下：

第一步　输入 x 与 y。

第二步　判断 x 是否为 0：

若 $x=0$，则输出错误信息；

否则计算 $y/x \Rightarrow z$，且输出 z。

（2）算法描述语言

为了说明程序的流程，还可以用专门规定的某种语言来描述，这种语言通常称为算法描述语言。算法描述语言一般介于自然语言与程序设计语言之间，它具有自然语言灵活的特点，同时又接近于程序设计语言的描述。但必须指出，用算法描述语言所描述的流程，一般不能直接作为程序来执行，最后还需转换成用某种程序设计语言所描述的程序。算法描述语言与

程序设计语言最大的区别就在于，算法描述语言比较自由，不像程序设计语言那样受语法的约束，只要描述的人们能理解就行，而不必考虑计算机处理时所要遵循的规定或其他一些细节。

在程序设计过程中，一般不可能开始就用某种程序设计语言来编制计算机程序，而是先用某种简单、直观、灵活的描述工具来描述处理问题的流程。当方案确定以后，再将这样的流程转换成计算机程序。这种转换往往是机械的，已经不涉及功能的重新设计或控制流程的变化，而只需考虑程序设计语言所规定的语法要求以及一些细节问题。

例如，在例 1.1 中，计算并输出 $z=y/x$，其处理的流程可以用一种算法描述语言描述如下：

```
INPUT  x, y
IF (x=0)  THEN
   OUTPUT  "ERROR"
ELSE
   { z=y/x
     OUTPUT  z
   }
```

对于这样的描述，只要懂一些英语，是不难理解的。有时为了使描述更简练，在算法描述语言中还可以使用一些自然语言。例如，上面的描述可以改为

```
输入  x, y
IF (x=0)  THEN
    输出错误信息
ELSE
   { z=y/x
     输出  z
   }
```

（3）流程图

人们在程序设计的实践过程中，总结出了一套用图形来描述问题的处理过程，使流程更直观，更易被一般人所接受。用图形描述处理流程的工具称为流程图。目前用得比较普遍的是结构化流程图，即 NS 图。在本书的 1.3 节中将介绍这种流程图。为了便于比较，先给出用结构化流程图描述计算并输出 $z=y/x$ 的处理流程，如图 1.2 所示。

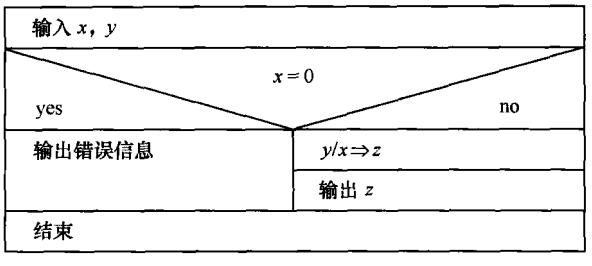


图 1.2　结构化流程图例

（4）编程

用某种程序设计语言编写的程序，本质上也是问题处理方案的描述，并且是最终的描述。但在一般的程序设计过程中，不提倡一开始就编写程序，特别是对于大型的程序。程序是程序设计的最终产品，需要经过中间每一步的细致加工才能得到。如果企图一开始就编写出程序，往往会

适得其反，达不到预想的结果。

下面是用 C 语言编写的计算并输出 $z=y/x$ 的程序：

```
#include <stdio.h>
void main()
{ float  x, y, z;
  printf("input  x, y: ");              /*输入提示*/
  scanf("%f, %f", &x, &y);              /*输入 x 与 y 的值*/
  if  (x==0)  printf("error! x=0\n"); /*若 x=0，则输出错误信息*/
  else                                  /*否则计算并输出结果*/
    { z=y/x; printf("z=%f\n", z);  }
}
```

5. 调试与运行

最后编写出的程序还需要进行测试和调试，只有经过调试后的程序才能正式运行。

所谓测试，是指通过一些典型例子，尽可能多地发现程序中的错误。因此，测试的目的是为了发现程序中的错误，而不是为了证明程序正确。

所谓调试，是指找出程序中错误的具体位置，并改正错误。因此，调试又称查错。

测试与调试往往是交替进行的，通过测试发现程序中的错误，通过调试进一步找出错误的位置并改正错误。这个过程需要重复多次。

1.3 程序设计的基本方法

程序设计方法是影响程序设计成败以及程序设计质量的重要因素之一。本节简要介绍以下 3 个方面：

（1）结构化程序设计；

（2）模块化程序设计；

（3）自顶向下、逐步细化的设计过程。

1.3.1 结构化程序设计

结构化程序设计要求把程序的结构限制为顺序、选择和循环 3 种基本结构，以便提高程序的可读性。这种结构化程序具有以下两个特点：

（1）以控制结构为单位，只有一个入口和一个出口，使各单位之间的接口比较简单，每个单位也容易被人们所理解。

（2）缩小了程序的静态结构与动态执行之间的差异，使人们能方便、正确地理解程序的功能。

在结构化程序设计过程中，经常使用的工具是 NS 图。

NS 图是一种不允许破坏结构化原则的图形算法描述工具，又称盒图。在 NS 图中，去掉了传统流程图中容易引起麻烦的流程线，全部算法都写在一个框内，每一种基本结构也是一个框。

NS 图有以下几个基本特点：

（1）功能域比较明确，可以从框图中直接反映出来。

（2）不可能任意转移控制，符合结构化原则。

（3）很容易确定局部和全程数据的作用域。

（4）很容易表示嵌套关系，也可以表示模块的层次结构。

下面简要介绍利用 NS 图描述的 3 种基本控制结构的形式。

1. 顺序结构

顺序结构的结构化流程图如图 1.3 所示。在图 1.3 中，块 S_1、S_2、S_3 是按顺序执行的。

2. 选择结构

选择结构分为两路分支选择结构和多路分支选择结构。其中多路分支选择结构又称为分情形选择结构。

（1）两路分支结构

两路分支选择结构的结构化流程图如图 1.4 所示。由图 1.4 可以看出，在两路分支选择结构中，当条件满足时，执行块 S_1，否则执行块 S_2，两者选一。

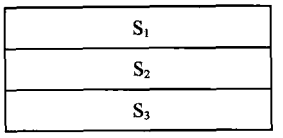


图 1.3　顺序结构的结构化流程图

（2）多路分支结构

多路分支选择结构的结构化流程图如图 1.5 所示。由图 1.5 可以看出，在多路分支选择结构中，根据条件的取值分为多种情况，不同情况将选择不同的操作。

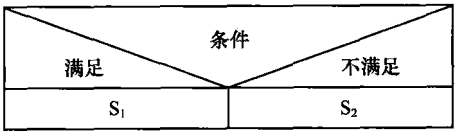


图 1.4　两路分支选择结构的结构化流程图

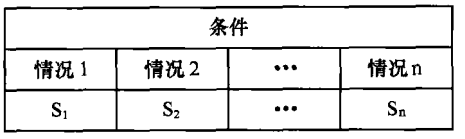


图 1.5　多路分支选择结构的结构化流程图

3. 循环结构

循环结构分为当型循环结构和直到型循环结构。

（1）当型循环结构

当型循环结构的结构化流程图如图 1.6 所示。由图 1.6 可以看出，在当型循环结构中，当条件满足时就执行块 S；否则退出循环，执行该循环结构后面的程序。显然，在当型循环结构中，块 S（称为循环体）有可能一次也不执行（即一开始条件就不满足）。值得指出的是，在循环体 S 中，应该要有改变条件的成分，否则将会造成死循环。

（2）直到型循环结构

直到型循环结构的结构化流程图如图 1.7 所示。由图 1.7 可以看出，直到型循环结构与当型循环结构的不同之处是，在直到型循环结构中，首先执行循环体 S，然后判断条件，若条件不满足，则继续执行循环体，这个过程直到条件满足，此时退出循环，执行该循环结构后面的程序。显然，在直到型循环结构中，循环体至少要执行一次。与当型循环结构一样，在直到型循环结构的循环体 S 中，也应该要有改变条件的成分，否则也将会造成死循环。

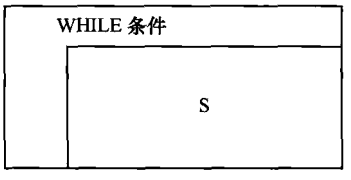


图 1.6　当型循环结构的结构化流程图

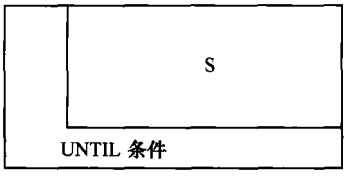


图 1.7　直到型循环结构的结构化流程图