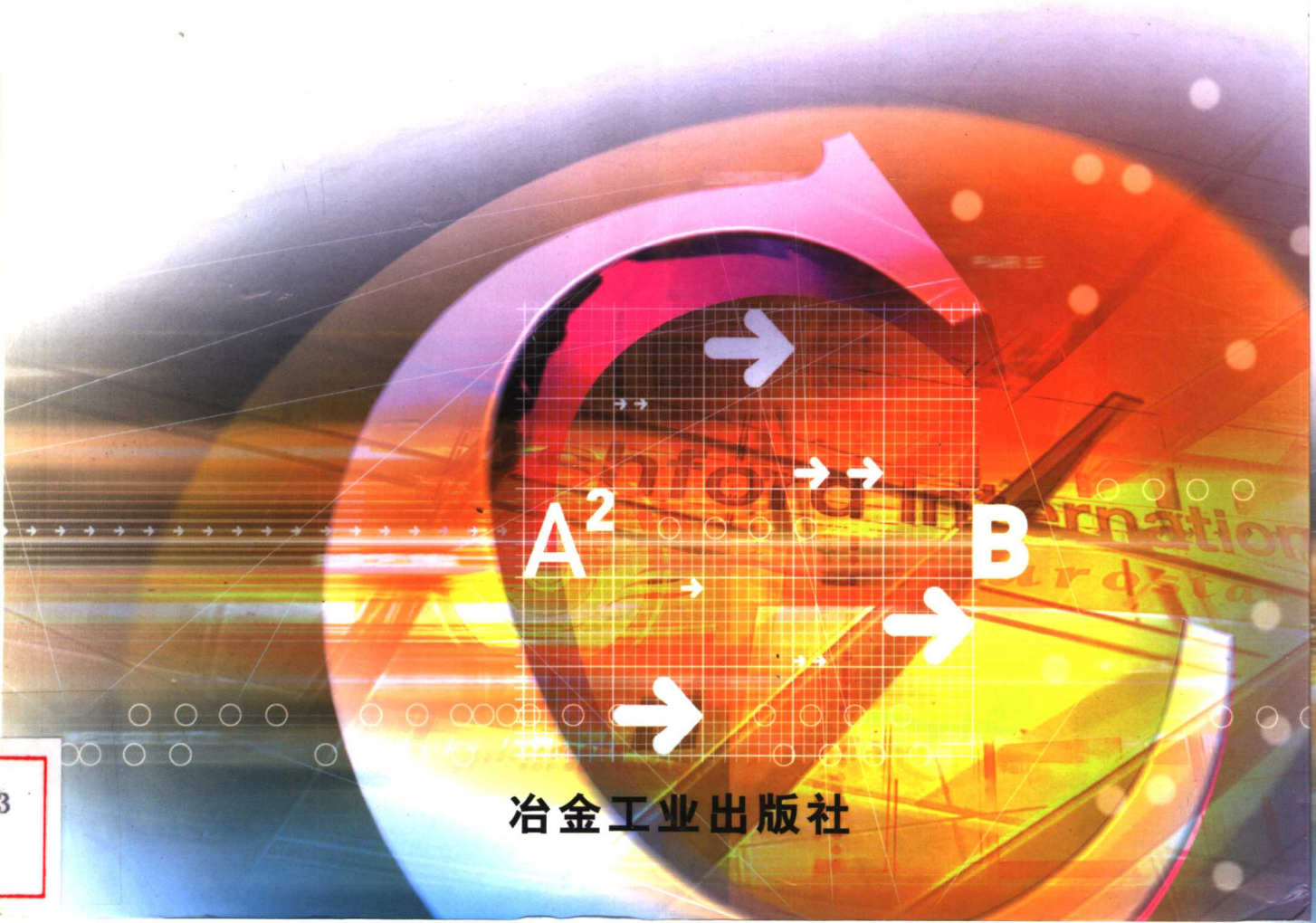
高等学校21世纪计算机教材

C语言程序设计及应用

谢乐军　编著



冶金工业出版社

高等学校21世纪计算机教材

C语言程序设计及应用

谢乐军　编著

北　京

冶金工业出版社

2004

内 容 简 介

本书全面、系统地介绍了C语言程序设计及应用的知识。包括：程序设计基础、C语言的基本概念、顺序结构程序设计、分支结构程序设计、循环结构程序设计、函数、指针、数组、结构体、共用体和枚举类型、编译预处理、位运算、文件、C语言的图形功能简介以及综合实例等内容。

本书结构清晰、语言通俗易懂、实例丰富，除14章外均配有同步练习题，最后还附有常用资料和习题答案，方便读者掌握和巩固所学知识。

本书适用面广，不仅可作为大中专院校相关专业和计算机培训班学习C语言程序设计的教材，还可供计算机工程技术人员参考。

图书在版编目（CIP）数据

C语言程序设计及应用 / 谢乐军编著. —北京：冶金工业出版社，2004.1
ISBN 7-5024-3461-5

I. C… II. 谢… III. C语言—程序设计
IV. TP312

中国版本图书馆CIP数据核字（2003）第120097号

出版人 曹胜利（北京沙滩嵩祝院北巷39号，邮编100009）
责任编辑 戈兰
湛江蓝星南华印务公司印刷；冶金工业出版社发行；各地新华书店经销
2004年1月第1版，2004年1月第1次印刷
787mm×1092mm 1/16； 19.5印张； 451千字； 304页； 1-3500册
25.00元
冶金工业出版社发行部 电话：（010）64044283 **传真：**（010）64027893
冶金书店 地址：北京东四西大街46号（100711） **电话：**（010）65289081

前　言

一、关于本书

C 语言是目前广泛应用的计算机程序开发语言，它不仅具有速度快、占用资源少等汇编语言的优点，还具有可移植性强、便于理解和掌握高级语言的优点。越来越多的计算机工程技术人员要掌握和应用 C 语言；越来越多的大中专院校的相关专业将 C 语言列为必修课程。

为此，我们根据近年来教学和科研的经验，结合 C 语言的技术特点，编写了本书。希望本书的问世，能满足广大计算机工程技术人员和大中专院校的学习 C 语言程序设计及应用的需要。

二、本书内容结构

本书共分 14 章，内容结构安排如下：

第 1 章：程序设计基础。本章介绍内容主要包括：程序设计基本概念、一般程序设计方法、结构化程序设计、程序设计风格、C 语言简介等内容。

第 2 章：C 语言的基本概念。本章介绍内容主要包括：常量与变量、简单数据类型、运算符与表达式、类型转换等内容。

第 3 章：顺序结构程序设计。本章介绍内容主要包括：顺序结构的设计思想、实现顺序结构的语句、顺序结构程序设计举例等内容。

第 4 章：分支结构程序设计。本章介绍内容主要包括：分支结构的设计思想、实现分支结构判断条件的构成、实现分支结构的语句、分支结构程序举例等内容。

第 5 章：循环结构程序设计。本章介绍内容主要包括：循环结构的设计思想、实现循环结构的语句、循环嵌套的概念及实现、循环结构的程序设计举例、其他语句等内容。

第 6 章：函数。本章介绍内容主要包括：函数基础、函数调用、函数的嵌套调用和递归调用、变量的作用域和存储类别、外部函数与内部函数、应用举例等内容。

第 7 章：指针。本章介绍内容主要包括：指针的概念、指针的变量、函数之间的地址传送、函数的指针与指向函数的指针变量、应用举例。

第 8 章：数组。本章介绍内容主要包括：一维数组、二维数组、字符数组、一维数组和指针、二维数组和指针、字符串和指针、向函数传递数组、指针数组及带参 main 函数、应用举例等内容。

第 9 章：结构体、共用体和枚举类型。本章介绍内容主要包括：结构体类型与结构体变量、结构体数组、向函数传递结构体数据、结构指针变量的说明和使用、链表、共用体、枚举类型、应用举例等内容。

第 10 章：编译预处理。本章介绍内容主要包括：宏定义、文件包含、条件编译、应用实例等内容。

第 11 章：位运算。本章介绍内容主要包括：位运算基础、位运算符、位域、应用实例

等内容。

第 12 章：文件。本章介绍内容主要包括：C 文件的基础知识、文件指针、文件操作、应用实例等内容。

第 13 章：C 语言的图形功能简介。本章介绍内容主要包括：C 语言图形模式的基本概念、基本图形函数、应用实例等内容。

第 14 章：综合实例。本章介绍内容主要包括：C 语言编程总结、应用实例等内容。

本书最后附有：

附录 A：标准库函数。

附录 B：ASCII 码表。

附录 C：C 语言常用关键字。

附录 D：C 语言运算符优先级别及结合方式。

附录 E：Turbo C（V2.0）编译错误信息。

附录 F：Turbo C（V2.0）使用指南。

第 1 ~ 13 章中的习题参考答案。

三、本书特点

本书结构清晰、语言通俗易懂、实例丰富，除第 14 章外均配有同步练习题，最后还附有相应的常用资料和习题答案，方便读者掌握和巩固所学知识。

四、本书适用对象

本书适用面广，不仅可作为大中专院校相关专业和计算机培训班学习 C 语言程序设计的教材，还可供计算机工程技术人员参考。

本书意在提供一本内容实用并具特色的 C 语言程序设计教材，但由于水平有限，难免有错误和不妥之处，希望广大读者批评指正。读者如果有好的意见或建议，可以发 E-mail 到 **service@cnbook.net**，也可以登录网站 **http://www.cnbook.net**，在该网站的论坛进行探讨。

编　者

2003 年 12 月

目　录

第 1 章　程序设计基础

在学习程序设计时，既要掌握所使用的某一种计算机语言（如 C 语言），更要掌握解题的方法和步骤，这是程序设计中的关键。语言只是一个人机交流的工具。本章将介绍算法、数据结构和程序设计方法，这是以后学习程序设计的基础，最后还简要地介绍一下 C 语言的概况。

1.1　程序设计基本概念

1.1.1　程序

人和计算机打交道，必须要解决一个语言沟通问题，因为计算机不能理解和执行人们使用的自然语言，而只能接受和执行二进制的指令。这些指令的集合就叫做“程序”（program）。换言之，一个程序是完成某一特定任务的一组指令序列，或者说，是为实现某一算法的指令序列。

1.1.2　算法

为解决一个问题而采取的方法和步骤，称为“算法”（algorithm）。或者说，算法是解题方法的精确描述。解决一个问题的过程就是实现一个算法的过程。对同一个问题，往往有不同的解题方法，各种方法有利有弊，为了有效地进行计算，就应当选择合适的算法。选择算法的主要标准是它的正确性和可靠性，简单性和易理解性，其次是所需要的存储空间少和执行快等。

一个算法应具有以下特点：

（1）有穷性。一个算法应该包含有限个步骤，而不能是无限的。

（2）确定性。算法的每一步都应当是明确无误的，不能含义模糊。

（3）有零个或多个输入。所谓输入是指执行指定的算法时，需要外界提供的信息。

（4）有一个或多个输出。没有输出的算法是没有意义的。

（5）有效性。算法中的每一步都应该能有效地执行。

为了使读者更好的理解什么是算法，下面举一个简单的例子来说明。

【例 1-1】求 n!（即求 n 的阶乘）。设 SUM 代表累乘之积，A 代表乘数。

（1）置 SUM=1，A=1。

（2）使 SUM *A，得到的结果放在 SUM 中。

（3）使 A 增加 1。

（4）如果 A≤n，跳到步骤（2）；如果 A>n，则算法终止，此时 SUM 中的值就是所求的结果。

显然，只要 n 大于等于零，这个算法就是正确的。

算法通常按某种规则来描述，常用的有：自然语言、传统流程图、伪代码和 N-S 结构流程图。

1. 自然语言

上面的【例 1-1】就是用自然语言描述的，自然语言的优点是通俗易懂，缺点是比较冗长、容易产生歧义，难以表达复杂问题，也难以转化为计算机语言。所以一般不用自然语言表示算法。

2. 传统流程图

传统流程图用不同的几何图形来代表各种不同性质的操作，直观清晰，曾经得到广泛的使用。在传统流程图中常用的符号包括：矩形的处理框、菱形的判断框、平行四边形的输入输出框和带箭头的流程线等，如图 1-1 所示。【例 1-1】的传统流程图表示如图 1-2 所示。

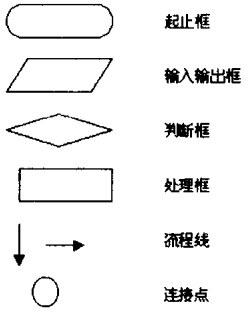


图 1-1 流程图符号

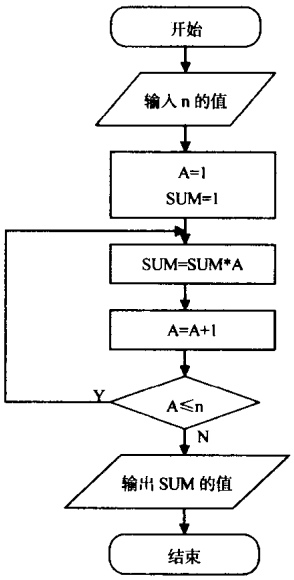


图 1-2 传统流程图

可以看到，用传统的流程图表示算法，逻辑清楚，形象化，容易理解。不过流程图占用的篇幅比较多，当算法比较复杂的时候，每一个步骤要画一个框，比较麻烦，而且结构不够清晰，画法太随意，所以现在它已经不如以前那样被普遍使用了。但还有许多计算机类的书刊使用这种流程图。

画流程图时，每个框内要说明操作内容，不要有歧义，不要忘记画箭头或把箭头画反了。

3. 伪代码

伪代码是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。它是一种混杂语言，使用一种结构化程序设计语言的语法控制框架，而在内部却可灵活地使用自然语言来表示各种操作条件和过程。伪代码与程序设计语言的区别在于，伪代码是不能被编译的。它不用图形，因此比较紧凑，也比较易懂，但不标准。上面的例子可以用伪代码如下表示：

```
输入 n
使 A 和 SUM 的值为 1
DO
```

```
SUM=SUM*A
A=A+1
UNTIL  A>n
输出 SUM 的值
```

但这并不是伪代码的惟一写法，其实伪代码比较灵活，没有严格的语法规则，现在很多书都用这种方法来表示算法。

4. N-S 结构流程图

1966 年 Bobra 和 Jacopini 提出了三种基本结构：

（1）顺序结构，如图 1-3 所示。

（2）选择结构，如图 1-4 所示。

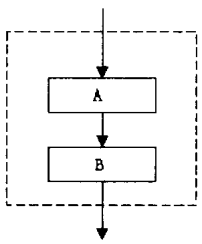


图 1-3　顺序结构

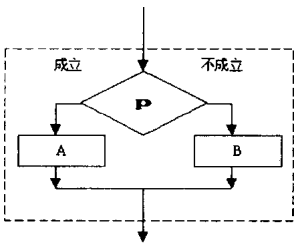


图 1-4　选择结构

（3）循环结构，如图 1-5 所示。

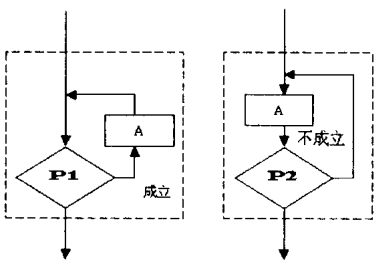


图 1-5　循环结构

分当型循环结构和直到型循环结构两种。

这三种结构有以下几个特点：

（1）只有一个入口。

（2）只有一个出口。

（3）结构内的每个部分都有可能被执行到。

（4）结构内没有死循环。

已经证明，由这三种基本结构组成的算法结构可以解决任何复杂的问题。由基本结构所构成的算法叫做结构化算法。

由于传统流程图画法太随意，对流程线的指向没有限制，因此当算法比较复杂的时候这种流程图就会变得难以阅读和修改。为此，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除了流程线，所有算法写在一个矩形框内，在框内还可以包含其他的框。这种流程图叫做 N-S 结构流程图（以二人名字的头一个字母组成）。

N-S 图由上面的三种基本结构组成：

（1）顺序结构，如图 1-6 所示。

（2）选择结构，如图 1-7 所示，表示当条件 p 成立时执行 A 操作，不成立时执行 B 操作。

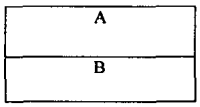


图 1-6 顺序结构

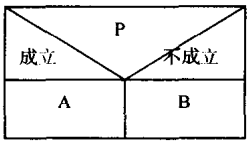


图 1-7 选择结构

（3）循环结构。以图 1-8 表示一个当型循环，如图 1-8 所示。以图 1-9 表示一个直到型循环，如图 1-9 所示。

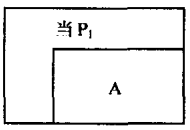


图 1-8 当型循环

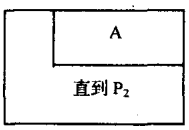


图 1-9 直到型循环

图 1-8 表示“当 P_1 条件成立时反复执行 A 操作”，即先判断，后执行。图 1-9 表示“反复执行 A，直到 P_2 条件满足为止”。

用以上三种基本框可以组成复杂的 N-S 结构化流程图。仍以【例 1-1】为例，其 N-S 图如图 1-10 所示。

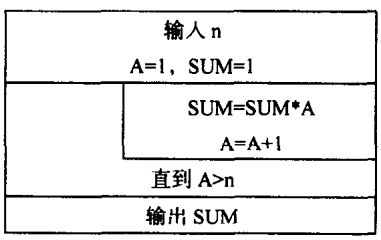


图 1-10 N-S 流程图

对照图 1-2，可以很清楚的看出 N-S 流程图的优越性。

1.1.3 数据结构

计算机算法的处理对象是描述客观事物属性的数据，由于客观事物的多样性，数据有不同的形式，如整数、实数、字符、图像、声音等。它是计算机程序加工的“原料”。数据结构是指数据对象及其相互关系和构造方法。一个数据结构 S 可以形式地用一个二元组表示：

S=（D，R）

其中，D 是数据结构中的数据（又称为“结点”的非空有限集合），R 是定义在 D 上的关系的非空有限集合。举例说明：

【例 1-2】在计算机科学中，复数可以如下定义：

COMPLEX=（C，R）

其中 C 是含两个实数的集合{c1，c2}；R={P}，而 P 是定义在集合 C 上的一种关系{<c1，c2>}，其中 c1 是复数的实部，c2 是复数的虚部。

结构是指结点之间的关系，“数据结构”就是结点的有限集合和关系的有限集合。数据结构中，结点及结点间的相互关系是数据的逻辑结构。数据结构在计算机中的表示称为数据的物理结构，包括数据元素的表示和关系的表示。

数据结构按逻辑关系的不同分为线性结构和非线性结构，其中非线性结构又可分为树形结构和图结构，而树形结构又可分为树结构和二叉树结构。

算法和数据结构之间存在密切关系，算法是建立在数据结构基础上的。未确定对数据进行如何操作就无法决定如何构造数据。同样，确定算法也依赖于作为基础的数据结构。明确了问题求解的算法，才能较好地设计数据结构。但要设计好的算法，又常常依赖于合理的数据结构，数据结构是设计算法的基础。对于一些复杂的问题，常有因数据结构的差异，问题的求解算法也会完全不同，设计合理的数据结构常可有效地简化算法。

要使计算机能完成人们预定的工作，首先必须为如何完成工作设计一个算法，然后再根据算法编写程序。计算机程序要对问题的每个对象和处理规则给出正确详尽的描述，其中程序的数据结构和变量用来描述问题的对象，程序结构、函数和语句用来描述问题的算法。算法和数据结构是程序的两个重要方面。

1.1.4 程序设计语言

程序语言可以分为低级语言和高级语言两大类。

低级语言又叫面向机器的语言，它是特定的计算机系统所固有的语言。它又可分为机器语言和符号语言（汇编语言）两类。

机器语言就是计算机能够直接识别和执行的指令集合。由于计算机只能识别“0”和“1”两种状态（即电气的正负状态），所以机器语言指令都是二进制指令。例如某种型号的计算机以 10000000 表示“进行一次加法”。很明显，这种由 0 和 1 组成的指令难学、难记、难阅读、难修改，给用户带来很大的不便。而且机器语言随不同机器而异，因此移植性很差。但其优点是执行速度最快。

符号语言是从机器语言发展演变而来的，它用一些“助记符号”来代替那些冗长的二进制指令。例如用 ADD 表示加法，SUB 表示减法，等等。显然，这种表示形式要比机器语言容易理解和使用。但是计算机并不能直接执行符号语言程序，必须先把它们逐条翻译成机器指令，然后交计算机执行。这个翻译工作叫“汇编”，是由一种专门的汇编程序来实现的，因此符号语言又叫汇编语言。虽然用汇编语言来编制程序，使编程的效率和程序的可读性提高了，然而，汇编语言始终是一种和机器语言非常接近的语言，它的书写格式在很大程度上取决于特定计算机上的机器指令，因此，它仍然是一种低级语言，对于人们的抽象思维和交流十分不便。

为了解决程序员的困难，高级语言发展起来了。50 年代出现了 FORTRAN 语言，60 年代又相继出现了 COBOL、ALGOL 60、BASIC 等语言。后来又陆续出现了 PASCAL、C、C++等。这些语言的特点是：用一种接近自然语言和数学语言的专用语言来表示算法，而且与具体的计算机无关，即用它所写的程序可以在任一种计算机上运行。高级语言的出现大大提高了程序设计的效率，使人们能更方便地使用计算机。

然而，今天的计算机仍然只能理解和执行机器语言。高级语言的引入意味着：必须有一个程序来使机器能理解用某一种高级语言编写的程序，担负这个任务的程序称为编译程

序。在一个计算机上运行某一种高级语言的前提是：该计算机系统配置了该语言的编译程序。以 C 语言为例，要运行 C 语言源程序，就必须安装 C 语言编译程序。而各种基于 C 语言的开发工具，如 TURBO C、BORLAND C、VISUAL C++等，都是内嵌了 C 语言的编译程序。

现将常见的高级语言介绍如下：

FORTRAN 语言是科学和工程算法中普遍使用的编程语言。FORTRAN 语言的主要缺点是不能直接支持结构化程序设计。

BASIC 语言简单易学，适用于小型事务处理程序。

COBOL 语言是商业数据处理中广泛使用的语言，由于它本身结构上的特点，使它能有效的支持与商业处理有关的、范围广泛的过程技术。它的缺点是比较繁琐。

PASCAL 语言是最早出现的一种结构化语言，具有丰富的数据类型、严谨的语法结构。PASCAL 语言已广泛用于科学、工程和系统程序设计中。

ADA 语言是一种工程化语言。ADA 语言在语言结构上是 PASCAL 型的，但它有完善的实时处理特性，包括中断处理、多重任务和进程通讯。

C 语言作为 UNIX 操作系统的主要使用语言，由于 UNIX 操作系统的成功，C 语言得到了广泛使用。C 语言具有很强的功能和高度的灵活性。和其他高级语言一样，C 语言能提供丰富的数据类型及丰富的供运算和数据处理用的运算符。

1.2 一般程序设计方法

对于一般的问题，设计一个程序大概要经过以下几个步骤：

（1）建立数学模型。

（2）选定算法，并用适当的工具描述。

（3）编程。

（4）测试及调试。

1.2.1 建立数学模型

建立数学模型是程序设计中最复杂、最困难的一步，好的数学模型本身就是一个定律，它要通过大量观察、分析、推理、验证等工作才可以得到。但这是数学范畴的工作，计算机工作人员一般不用去完成这部分的研究工作，但也应该对其基本知识有一定了解。在进行程序设计时往往都是利用已有的基本数学模型去构造出问题的模型。各种数学，如微积分、运筹学、图论、高等数学等都是不同的基本数学模型。

1.2.2 选定算法

算法是解决问题的方法与步骤。设计算法也是一件非常困难的工作，经常采用的算法设计技术主要有迭代法、穷举搜索法、递推法、贪婪法、回溯法、分治法、动态规划法等等。另外，为了以更简洁的形式设计和描述算法，在设计算法时又常常采用递归技术，用递归描述算法。

解决一个问题往往有多种算法，一个算法的好坏将直接影响到一个程序的质量。通常设计一个好的算法应考虑达到以下要求：

（1）正确性。算法应当满足具体问题的需求。

（2）可读性。算法主要是为了人的阅读与交流，其次才是机器执行。可读性好有助于人对算法的理解，而晦涩难懂的程序易于隐藏较多错误难以调试和修改。

（3）健壮性。它是指当输入非法数据时，算法也能适当地作出反应或进行处理，而不会产生莫名其妙的结果。

（4）效率与存储空间需求。效率指的是算法执行时间。对于同一个问题如果有多个算法可以解决，执行时间短的算法效率高。存储空间需求指算法执行过程中所需要的最大存储空间，显然，空间需求较小的算法质量更高。

举个例子，对于日常工作中最普通的排序问题，就有若干种解决算法，如：直接插入排序、2-路插入排序、希尔排序、冒泡排序等等。各种算法都有不同的时间和空间要求，在实际应用中应该具体问题具体分析。

1.2.3　编程

编程就是将选定的算法从非计算机语言的描述形式转换为计算机语言的语句形式描述出来。这个过程和特定的高级语言有关，同一个算法可以用不同的高级语言来实现。因此，在编程前首先要选择开发的语言。每种语言都有自己的特点，为一个特定的项目选择语言时通常可以考虑以下因素：

（1）应用领域。

（2）算法和计算的复杂性。

（3）软件运行的环境（包括可使用的编译程序）。

（4）用户需求（特别是性能需求）。

（5）数据结构的复杂性。

（6）开发人员的水平。

从应用领域来看，COBOL 语言适用于商业领域的应用，FORTRAN 语言适合于科学和工程计算的应用，PROLOG 和 LIST 适用于人工智能的应用，对于一些采用面向对象方法的应用系统通常可采用 Smalltalk 或 C++。有些语言可用于多种应用领域，例如 C 语言，它原先是为辅助开发 UNIX 操作系统而设计的，主要用于开发系统软件，现在已广泛应用于其他领域。

开发和维护高级语言程序比开发和维护低级语言程序要容易得多，但是高级语言程序经编译后所产生的目标程序要比完成相同功能的低级语言程序长得多，也就是说前者的功效要比后者的功效低。因此某些对运行时间或存储空间有过高要求的项目，或者在某些不能提供高级语言编译程序的计算机上开发程序（如单片机），往往要使用低级语言。在某些大型系统中，为了提高系统的运行效率，有时也对系统中在执行时间上起关键作用的模块局部使用低级语言。

选择了语言以后，还要灵活运用高级语言的特性来解决现实问题。在程序设计中要特别注意以下三点：

（1）语法。每种语言都有自己的语法规则，这些规则是非常严格的。在进行编译时系统会按语法规则严格检查程序，如有不符合语法规则的地方，计算机会显示语法有错信息。

（2）语义。即某一语法成分的含义。例如，C语言中用“int”定义整型变量，用“char”定义字符型变量，用“while”语句实现循环，用“*”表示乘法，用“！=”表示不等运算等等。在使用时必须正确了解每一种语法成分的正确含义。

（3）语用。即正确使用语言。要善于利用语法规则中的有关规定和语法成分的含义有效地组成程序以达到特定的目的。

1.2.4 测试及调试

编程完成以后，首先应该静态审查程序，即由人工“代替”或“模拟”计算机，对程序进行仔细检查，然后将高级语言源程序输入计算机，经过编译、连接，然后运行。在编译、连接及运行时如果在某一步发现错误，就必须找到错误并改正，然后再从重新编译运行，直到得到正确结果为止。

程序测试的目的是尽可能多地发现程序中的错误和缺陷。要进行测试，除了要有测试数据外，还应同时给出该组测试数据应该得到怎样的输出结果。在测试时将实际的输出结果与预期结果比较，若不同则表示发现了错误。对非法的和非预期的输入数据也要像合法的和预期的输入数据一样进行测试。另外，还要检查程序是否做了不应该做的事。

1.3 结构化程序设计

一般来说，从实际问题抽象出数学模型是有关领域的专业工作人员的任务。程序设计人员的工作，最关键的一步就是设计算法。如果算法正确，将它转换为任何一种高级语言程序，并不困难。程序设计人员水平的高低在于他们能否设计出好的算法。程序质量主要由算法决定。早期的程序以追求效率为主要目标，往往不注意程序的可读性，程序设计无章可循，有时为了追求效率方面的微小改进，把程序改得晦涩难懂，增加了程序设计、调试和维护过程中的困难，程序的可靠性差。现代科学技术的发展，要求软件的生产方式从“个体方式”中解放出来，按照“工程”的方法来组织软件的生产，也就是说，按照一定的规范、一定的步骤来进行程序设计，而不允许程序设计人员随便地写程序。

前面已经提到过，只用三种基本的控制结构，通过组合和嵌套就能实现任何单入口单出口的程序——这就是结构化程序设计基本原理。这三种结构是顺序结构、选择结构和循环结构。使用结构化程序设计技术不仅能显著提高软件的生产率，而且可以保证获得结构清晰、易于测试、修改和验证的高质量的程序。

要设计出结构的程序，应当采用以下的方法：

（1）自顶向下。

（2）逐步细化。

（3）模块化。

所谓“自顶向下，逐步细化”，是指一种先整体后局部的设计方法。对于一个较复杂的问题，一般不能立即写出详细的算法或程序，但可以很容易写出一级算法，即求解问题的轮廓，然后对这个算法逐步求精，把它的某些步骤扩展成更详细的步骤，细化过程中，一方面加入详细算法，一方面明确数据，直到根据这个算法可以写出程序为止。自顶向下，逐步求精的方法符合人类解决复杂问题的思维方式，用先全局后局部，先整体后细节，先抽象后具体的过程开发出的程序层次结构清晰，容易阅读、理解和测试。