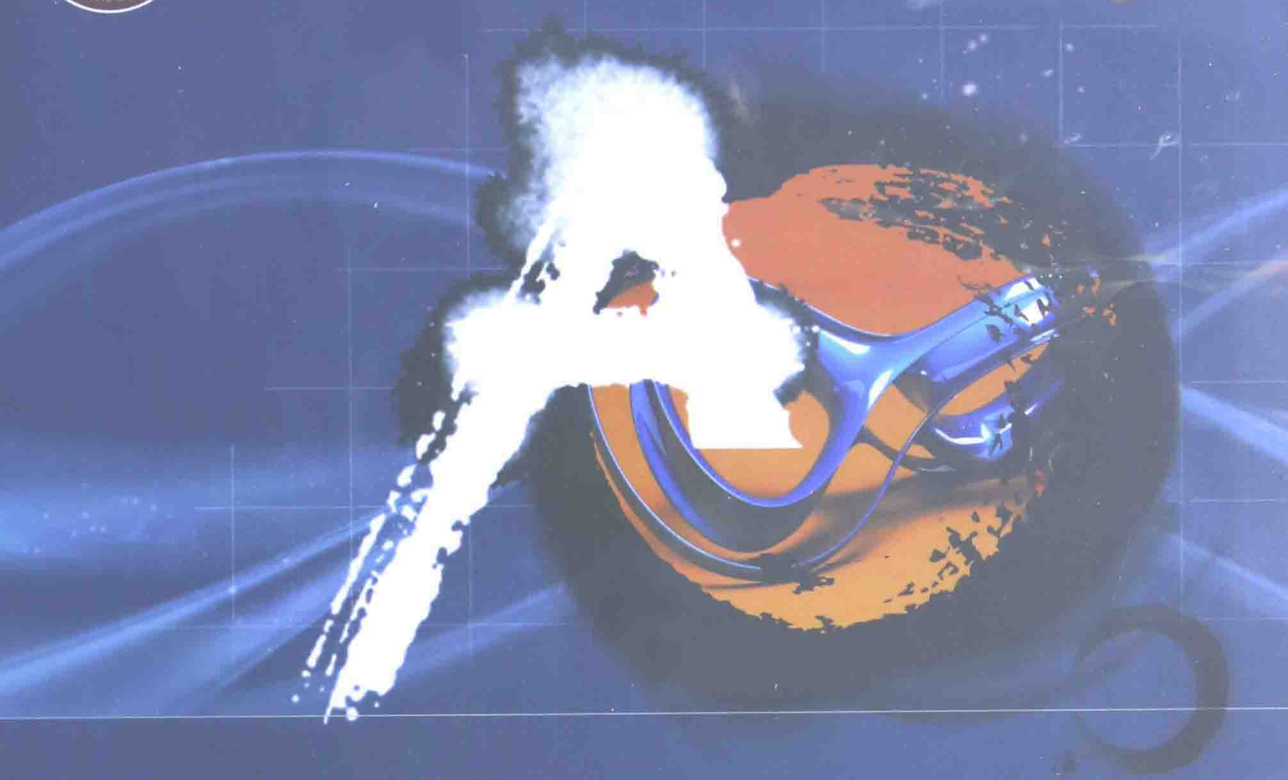
北京高等教育精品教材

BEIJING GAODENG JIAOYU JINGPIN JIAOCAI



高等学校规划教材

C++与数据结构

（第3版）

◎高 飞 薛艳明 主编 ◎白 霞 聂 青 副主编

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

高等学校规划教材
北京高等教育精品教材
国家精品课程教材

C++与数据结构

（第3版）

高　飞　薛艳明　主　编
白　霞　聂　青　副主编

電子工業出版社
Publishing House of Electronics Industry

内 容 简 介

本书是国家级（网络教育）精品课程的教学成果，也是北京市高等教育精品教材，根据教育部计算机基础课程教学指导委员会《计算机基础课程教学基本要求》中，有关理工类专业“算法基础与程序设计”课程要求组织编写。本书内容由浅入深，案例丰富，通俗易懂，实用性强。

本书在介绍了 C++语言的程序设计方法的基础上，采用面向对象的思想和抽象数据类型的概念，用C++语言有效地组织和描述了线性表、堆栈、队列、树和图等各种典型的数据结构和相关类的实现，并介绍了每种数据结构的不同存储方法、典型操作及其应用。

全书共 10 章，包括数据结构的基本概念，数组与指针，C++编程基础，STL 标准模版库，线性表，堆栈、队列和递归，树与二叉树，图，查找与散列结构，排序等。本书各章配有习题和实验训练题，方便实践教学，并为任课教师提供电子课件和示例源代码。

本书可作为高等院校电子信息类以及其他相关专业本科生教材和教学参考书，也可供从事程序设计工程人员参考使用。

图书在版编目（CIP）数据

C++与数据结构 / 高飞，薛艳明主编. —3 版. —北京：电子工业出版社，2015.3
ISBN 978-7-121-25683-7

Ⅰ. ①C… Ⅱ. ①高… ②薛… Ⅲ. ①C 语言—数据结构—程序设计—高等学校—教材 Ⅳ. ①TP312 ②TP311.12

中国版本图书馆 CIP 数据核字（2015）第 047745 号

策划编辑：袁　玺
责任编辑：袁　玺
印　　刷：涿州市京南印刷厂
装　　订：涿州市京南印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：19.25　字数：492.8 千字
版　　次：2015 年 3 月第 1 版
印　　次：2015 年 3 月第 1 次印刷
定　　价：40.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：(010) 88258888。

前　言

本书是**北京市高等教育精品教材**，编写者是**国家级优秀教学团队**和**北京市优秀教学团队“计算机公共课教学团队”**的主要成员。教材的编写以教育部计算机基础课程教学指导委员会《计算机基础课程教学基本要求》中，有关理工类专业“算法基础与程序设计”课程要求为指导思想，结合教学改革实践，总结了国家级精品课程和北京市精品课程“数据结构与算法设计”的建设经验，坚持以培养学生解决实践问题的能力为特色，以适应信息时代的人才培养模式。

♦ 本书的写作特点

数据结构是计算机算法设计的基础，在计算机科学中占有非常重要的地位。深入研究数据结构对构造完美算法结构和设计具有重要的作用。程序设计语言是实现算法的载体，语言只有满足算法实现的需求，才能被认识和掌握，数据结构只有通过程序语言才能在应用中发挥作用。因此，本书力求以算法为中介，使学习程序设计语言和学习数据结构共进。

本书在介绍了 C++语言的程序设计方法的基础上，采用 C++程序设计语言描述算法。C++是一种既支持面向过程程序设计，又支持面向对象程序设计的混合型语言，它独特的面向对象特征，可以为面向对象技术提供全面支持，是描述算法的一种较为理想的语言。采用面向对象程序设计语言描述施加与数据结构之上的算法，不仅有利于将面向对象技术相结合，也为上机实践提高高级语言程序设计水平提供了方便。

本书是在对《C++与数据结构（第 2 版）》进行精简和修改基础上完成的，共包括 10 章。第 1 章是对数据结构和面向程序设计方法的概述。从第 2 章到第 4 章，是 C++语言程序设计基础。主要内容是 C++编程基础，精炼地介绍了数组与指针、函数、C++类及其对象的封装性、引用、友元和重载、继承与派生、多态性与虚函数、模板以及 STL 的相关内容。从第 5 章到第 10 章，重点介绍典型的数据结构。主要内容包括：线性表、堆栈与队列、树与二叉树、图、查找与散列结构、排序。全书每章都配有习题以及相应的程序例题和实验题目。

♦ 本书的编排特点

本书每章均安排有**习题和实验训练题**，可方便实践教学。

对书中重要内容采用黑体标注。

本书**强调可读性**。书中程序全部采用统一的设计风格。例如，类名和变量名的定义做到“望名知义”；采用缩进格式组织程序；对程序中的语句尽可能多地使用注释。

本书包含大量的程序示例，给出了运行结果。凡是程序开头带有程序名编号的程序，都是可以直接在计算机中编译运行的程序。对于各章实验训练中的源代码，读者**可扫描相应的二维码**进行下载。

◆ 教学支持

本书的电子教案可以在讲课时用多媒体投影演示。教师不仅可以使用本教案，还可以方便地修改和重新组织其中的内容以适应自己的教学需要。使用本教案可以减少教师备课时编写教案的工作量，从而通过教学效果和效率。

本书向使用本教材的教师免费提供电子教案和程序示例代码。需要的教师可以直接登录华信教育资源网 http://www.hxedu.com.cn 免费注册下载。

本书由高飞设计总体架构。第 1，2 章由薛艳明编写，第 3，4 章由白霞编写，第 5，6，7 章由高飞编写，第 8，9，10 章由聂青编写。全书由高飞、薛艳明统稿和定稿。

在本书的编写过程中，北京理工大学信息与电子学院计算机教学团队中许多从事教学工作多年的同仁提出了许多宝贵的建议。电子工业出版社章海涛和袁玺编辑对本书的编写提出了宝贵的意见，在此对他们的辛勤工作和热心帮助一并表示衷心的感谢。

由于计算机算法和程序设计技术发展迅速，作者水平有限，书中的疏漏与不足在所难免，敬请广大读者和同仁不吝赐教，拔冗指正。

感谢您选用本书，欢迎您对本教材提出意见和建议。作者 E-mail：gaofei@bit.edu.cn

作者

于北京理工大学

2014 年 10 月

目　录

第 1 章　数据结构的基本概念

本章学习目标

通过对本章内容的学习，学生应该能够做到：

1. 了解：数据结构在计算机数据处理中的作用；基于面向对象描述数据结构算法的优势，以及“对象=数据结构+算法”的概念。

2. 理解：数据结构研究的数据之间的逻辑关系、数据在计算机内部的存储结构，以及在数据的各种结构上实施有效操作或处理（算法）等概念和相互关系。

3. 掌握：数据结构的相关基本概念与术语；面向对象的基本概念和基本思想。

计算机已经深入到人类社会的各个领域，计算机的应用已不再局限于科学计算，而是更多地用于控制、管理及数据处理等非数值计算的处理工作。与此相应，计算机加工处理的对象由纯粹的数值发展到字符、表格和图像等各种具有一定结构的数据，这就给程序设计带来一些新的问题。为了编写出一个好的程序，必须分析待处理对象的特性以及它们之间存在的关系，这就是“数据结构”这门学科形成和发展的背景。分析数据对象之间的逻辑关系，并用计算机存储结构体现出这些逻辑结构并操作这些数据，就是数据结构这门课程要解决的问题。

1.1　数据结构的概念和术语

在讨论数据结构之前，让我们先掌握几个与数据结构密切相关的概念和术语。

数据（Data）：数据是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机处理的符号的总称。它是信息的载体，是计算机程序加工的“原料”。对计算机科学而言，数据的含义极为广泛。一般来说，数据主要有两大类：一类是数值数据，包括整数、实数、复数等，主要用于工程、科学计算和商业事务处理；另一类是非数值数据，主要包括字符、字符串、图像、声音等，它们可以通过编码而转变为可被计算机处理的数据。

数据元素（Data Element）：数据元素是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。一个**数据元素**可由若干个**数据项**组成，例如，一个学生的基本信息为一个数据元素，可以包括学号、姓名、性别、年龄、成绩、家庭地址等多个数据项。数据项是数据处理中不可分割的最小单位。

数据对象（Data Object）：数据对象是性质相同的数据元素的集合，是数据的一个子集。例如：英文字母数据对象可以是集合 L={ ‘A’，‘B’，‘C’，…，‘Z’ }，整数数据对象可以是集合 N={–32767，–32766，…，–1，0，1，2，…，32768}。

数据结构（Data Structure）：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。在任何问题中，数据元素都不是孤立存在的，在它们之间存在某种关系，这种

数据元素相互之间的关系称为**结构**。

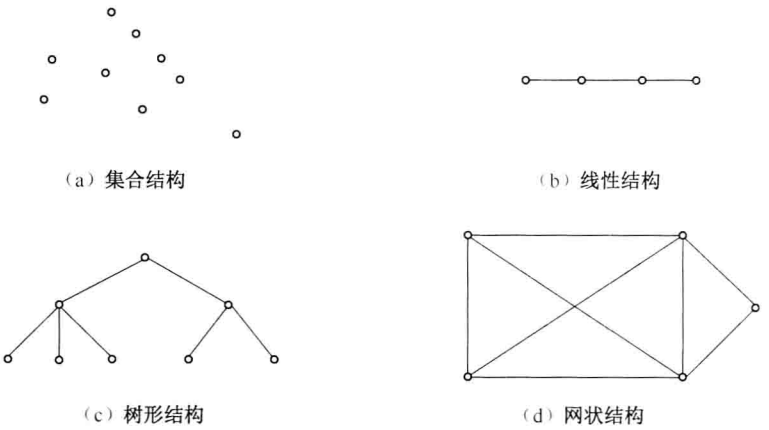
根据数据元素之间关系的不同特性，通常有如图 1.1.1 所示四类基本结构。

（1）**集合**　这种结构中的数据元素是无序且没有重复的元素，它们之间除了“同属于一个集合”的关系外，无其他关系。

（2）**线性结构**　这种结构中的数据元素之间存在一个对一个的关系，所有的数据成员按某种次序排列在一个序列中，除第一个元素外，每个元素都有一个且仅有一个直接前驱，第一个数据元素没有直接前驱；除最后一个元素外，每个元素都有一个且仅有一个直接后继，最后一个数据元素没有直接后继。

（3）**树形结构**　这种结构中的数据元素之间存在一个对多个的关系。

（4）**图状结构或网状结构**　这种结构中的数据元素之间存在多个对多个的关系。



（a）集合结构　（b）线性结构　（c）树形结构　（d）网状结构

图 1.1.1　四类基本结构

数据结构是一个二元组：

$$\text{Data\_Structure} = (D, S)$$

其中，D 是数据元素的有限集，S 是 D 上关系的有限集。

上述结构定义中，关系描述的是数据元素之间的逻辑关系，因此，又称为数据的**逻辑结构**。然而，讨论数据结构的目的是为了在计算机中实现对它的操作，因此，我们还需研究如何在计算机中表示它。

数据结构在计算机中的表示称为数据的**物理结构**，又称为**存储结构**，它包括数据元素的表示和关系的表示。在计算机中表示信息的最小单位是二进制的一位，叫作位。在计算机中，我们可以用一个由若干位组合起来形成的一个位串表示一个数据元素，通常称这个位串为**元素或结点**。当数据元素由若干数据项组成时，位串中对应于各个数据项的子位串称为**数据域**。因此，元素或结点可看成是数据元素在计算机中的映象。

数据元素之间的关系在计算机中有两种不同的表示方法：**顺序映象和非顺序映象**，对应两种不同的存储结构，即**顺序存储结构和链式存储结构**。顺序映象的特点是：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。非顺序映象的特点是：借助指示元素存储地址的指针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。一个算法的设计取决于问题的逻辑结构，而算法的实现依赖于采用的存储结构。

通常我们讨论数据结构，不但要讨论各种在解决问题时可能遇到的典型的逻辑结构，还要讨论这些逻辑结构的存储映象（存储实现），此外还要讨论这种数据结构的相关操作及

其实现。因此，数据结构要研究的主要内容可以简要地归纳为以下三个方面：

（1）研究数据之间固有的客观联系（**逻辑结构**）；

（2）研究数据在计算机内部的存储方法（**存储结构**）；

（3）研究如何在数据的各种结构上实施有效的操作或处理（**算法**）。

1.2 抽象数据类型

数据结构要研究逻辑结构和存储结构，那么，如何描述存储结构呢？存储结构涉及数据元素及其关系在存储器中的物理表示，由于本书是在高级语言的层次上讨论数据结构，因此不能直接用内存地址来描述存储结构，我们可以借用高级程序语言中提供的数据类型来描述它。例如，可用一维数组类型来描述顺序存储结构，用指针来描述链式存储结构。让我们首先回顾一下什么是数据类型。

1.2.1 数据类型

数据类型是一组性质相同的值的集合以及定义于这个值集合上的一组操作的总称。计算机处理的数据是以某种特定的形式存在的，例如，整数、浮点数、字符等形式。C++语言中可以使用的数据类型，如图 1.2.1 所示。

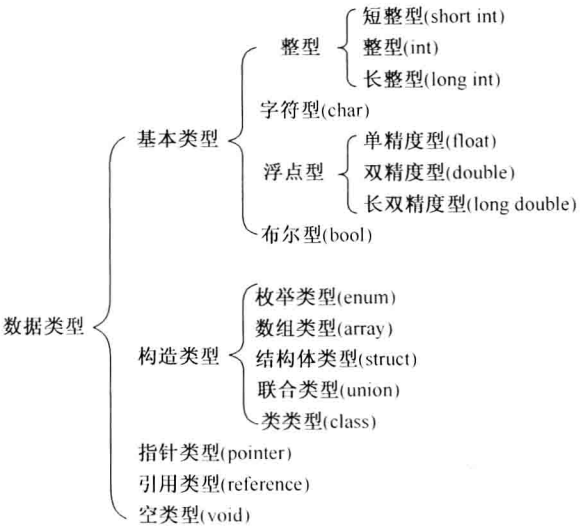


图 1.2.1 C++语言中的数据类型

C++语言中没有统一规定各类数据的精度、数值范围和在内存中所占的字节数，计算机所能表示的实际数据范围根据编译器和计算机系统结构不同而不同。表 1.2.1 是 Visual C++中数值型和字符型数据在内存中所占的字节数和数值范围。

表 1.2.1 数值型和字符型数据的字节数和取值范围

类　型	类型标识符	字　节	数 值 范 围
整型	[signed] int	4	−2147483648～+2147483647
无符号整型	unsigned [int]	4	0～4294967295
短整型	short [int]	2	−32768～+32767
无符号短整型	unsigned short [int]	2	0～65535
长整型	long [int]	4	−2147483648～+2147483647
无符号长整型	unsigned long [int]	4	0～4294967295
字符型	[signed] char	1	−128～+127
无符号字符型	unsigned char	1	0～255
单精度型	float	4	3.4×10^{-38}～$3.4\times10^{+38}$
双精度型	double	8	1.7×10^{-308}～$1.7\times10^{+308}$
长双精度型	long double	8	1.7×10^{-308}～$1.7\times10^{+308}$

（1）整型数据有长整型（long int）、整型（int）和短整型（short int）之分。C++中没有规定每种数据所占的字节数，只规定 int 型数据所占的字节数不大于 long 型数据，不小于 short 型数据。在 16 位机的 C++系统中，short 型数据和 int 型数据占 2 字节，long 型数据占 4 字节；在 32 位机的 C++系统中，short 型数据占 2 字节，int 型数据和 long 型数据占 4 字节。

（2）整型数据以二进制数的形式存储，例如，十进制数 65 的二进制数为 01000001，在内存中的存储形式如图 1.2.2 所示。

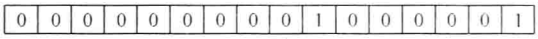


图 1.2.2　整型数据的存储形式

（3）整型数据和字符型数据都有带符号和无符号两种形式，分别由修饰符 signed 和 unsigned 表示。如果指定为 signed，则数值以补码形式存放，存储单元最高位表示数值的符号。如果指定为 unsigned，则数值没有符号，全部二进制数都用来表示数值本身。占 2 字节带符号短整型和无符号短整型的存储情况如图 1.2.3 所示。

图 1.2.3　有无符号短整型数据的存储情况

（4）浮点数有单精度（float）、双精度（double）和长双精度（long double）之分。Visual C++6.0 中，float 型有 6 位有效数字，double 型有 15 位有效数字。float 型占 4 字节，double 和 long double 型占 8 字节。

数据类型不但规定了使用该类型时的取值范围，而且还规定了该类型可以用的一组操作。例如，与整型有关的操作有+、–、*、\、%等。C++不但定义了一些基本的数据类型，还提供了复合的数据类型，如数组、结构体、共用体、类等。程序员可以利用这些复合的数据类型，自定义一些实际所需要的数据类型，例如，程序员可自定义线性表类、栈类等数据类型。

1.2.2　数据抽象与抽象数据类型

在面向对象的程序设计中，常常提到“抽象”一词，那么，什么是抽象呢？抽象的本质就是抽取反映问题本质的东西，忽略非本质的细节。

抽象数据类型通常是指由用户定义，用来表示应用问题的**数据模型**。抽象数据类型由基本的数据类型组成，并包括一组相关的操作，抽象数据类型类似于 C++中的类。对于一个数据成员完全相同的数据类型，如果给它定义不同的功能，则可形成不同的抽象数据类型。

抽象数据类型的特点是**使用与实现分离**，实行封装和信息隐蔽。在抽象数据类型设计时，应把类型的声明与其实现分离开来。首先根据问题的要求，定义该抽象数据类型需要包含哪些信息，并根据功能确定公共界面的服务，使用者可以使用公共界面的服务对该抽象数据类型进行操作。另一方面，抽象数据类型的具体实现作为私有部分封装在其实现模块内，使用者不能看到，也不能直接操作该类型所存储的数据，只能通过界面中的服务来访问这些数据。

从实现者的角度来看，把抽象数据类型的具体实现封装起来，有利于编码、测试，也有利于将来的修改。因为这样做可以使得错误局部化，一旦出现错误，其传播范围不至于影响其他模块，如果为了提高效率希望改进数据结构，只需要改变抽象数据类型的具体实现即可，只要界面中的服务的使用方式不变，其他所有使用该抽象数据类型的程序都可以不变，从而大大提高了系统的稳定性。

从使用者的角度来看，只要了解该抽象数据类型的规格说明，就可以利用其公共界面中的服务来使用这个类型，而不必关心其物理实现，这样使用者可以在开发过程中抓住重点，集中精力考虑如何解决应用问题，使问题得到简化。例如，我们在求解一个最优化问题时常常要使用一个栈，那么，应当首先考虑此栈应存放什么信息，应如何组织，至于栈如何实现，可能会出现哪些例外情况，这些例外情况如何处理等可以忽略，直接调用堆栈类提供的相关服务即可。

1.3 算法和算法分析

1.3.1 算法

数据结构除了要研究数据的逻辑结构和存储结构外，还要研究如何在数据的各种结构上实施有效的操作或处理，这就涉及算法。算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每条指令表示一个或多个操作。一个算法应具有下列五个重要特性。

1. 有穷性

对任何合法的输入值，一个算法必须在执行有穷步之后结束，且每步都应该在有穷时间内完成。

2. 确定性

算法中每条指令必须有确切的含义，读者理解时不会产生二义性，在任何条件下，算法只有唯一的一条执行路径，对于相同的输入只能得出相同的输出。

3. 可行性

算法必须是可行的，即算法中描述的操作都可以通过已经实现的基本运算执行有限次来实现。

4. 输入

一个算法有 0 个或多个输入，这些输入取自于某个特定的数据对象的集合，它可以使用输入语句从外部提供，也可以在算法内通过赋初值给定。

5. 输出

一个算法有一个或多个的输出，这些输出是与输入有着某些特定关系的量。

1.3.2 算法设计的要求

通常一个好的算法应考虑达到以下目标。

1. 正确性

算法应当满足具体问题的需求。通常一个大型问题的需求，要以特定的规格说明方式

给出，而一个实习问题或练习题，往往就不那么严格。目前大多采用自然语言描述需求，应当包括对于输入、输出和加工处理等明确的无歧义性的描述。设计或选择的算法应当能满足这种需求。正确性大体可分为以下四个层次：

（1）程序不含语法错误；

（2）程序对于输入的数据能够得出满足规格说明要求的结果；

（3）程序对于精心选择的、典型的、苛刻的输入数据能够得出满足规格说明要求的结果；

（4）程序对于一切合法的输入数据都能产生满足规格说明要求的结果。

要达到第四层含义下的正确是极为困难的，所有不同输入数据的数据量非常大，逐一验证的方法是不现实的。因此，对于大型软件需要进行专业测试，而在一般情况下，通常以第三层含义的正确性作为衡量一个程序是否合格的标准。

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，晦涩难懂的算法易于隐藏较多错误，往往难以调试和修改。

3．健壮性

当输入数据非法时，算法也能适当地做出反应或进行处理，而不会产生莫名其妙的结果，处理错误的方法可以是返回一个表示错误或错误性质的值。

4．效率

效率指的是算法执行时计算机资源的消耗，它包括运行时间代价和存储空间代价。对于同一个问题，如果有多个算法可以解决，那么执行时间短、存储量需求小的算法效率高。效率与问题的规模和性质有关，求 10 个人的平均工资与求 1000 个人的平均工资所花的执行时间或运行空间显然有一定的差别。接下来，我们将重点讨论算法的效率。

1.3.3 算法效率的度量

算法的效率包括算法运行时间代价和存储空间代价，它们分别由**时间复杂度**和**空间复杂度**来度量。

1．时间复杂度

算法执行时间需依据该算法编制的程序在计算机上运行时所消耗的时间来度量。度量一个程序的执行时间通常有两种方法，即事后统计方法和事前分析估算方法。

（1）**事后统计方法**　因为很多计算机内部都有计时功能，有的甚至可精确到毫秒级，不同算法的程序可通过一组或若干组统计数据来分辨优劣。但这种方法有两个缺陷，一是必须先运行依据算法编制的程序；二是统计数据依赖于计算机的硬件、软件等环境因素，有时容易掩盖算法本身的优劣。因此，人们常常采用另一种事前分析估算的方法。

（2）**事前分析估算方法**　用高级程序语言编写的程序在计算机上运行时所消耗的时间取决于下列因素：

① 问题的规模；

② 算法选用的策略；

③ 书写程序的语言，对于同一个算法，实现语言的级别越高，执行效率就越低；

④ 编译程序所产生的机器代码的质量；

⑤ 机器执行指令的速度。