普通高等教育“十一五”国家级规划教材

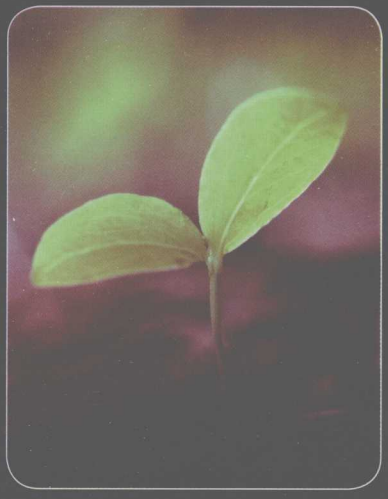
江苏省高等学校精品教材

数据结构
——使用C++语言描述（第2版）

Data Structures in C++ (2nd Edition)

陈慧南 主编

- 精心选材——难易适中、保留经典、拓展新知
- 适于学教——编制精良、示例丰富、讲解通俗
- 注重实践——完整C++算法实现，独立实习章节



精品系列

人民邮电出版社
POSTS & TELECOM PRESS

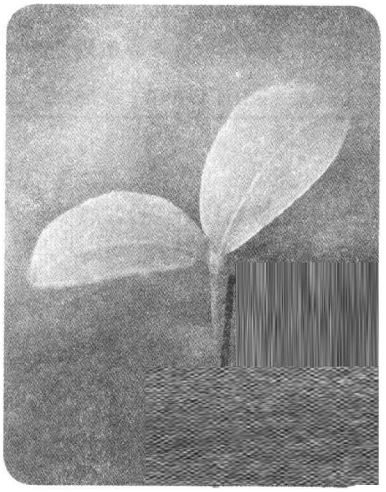
普通高等教育“十一五”国家级规划教材

江苏省高等学校精品教材

数据结构
——使用C++语言描述（第2版）

Data Structures in C++ (2nd Edition)

陈慧南 主编



精品系列

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

数据结构：使用C++语言描述 / 陈慧南主编. —2版.
北京：人民邮电出版社，2008.10
普通高等教育“十一五”国家级规划教材
21世纪高等学校计算机规划教材
ISBN 978-7-115-18402-3

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

中国版本图书馆CIP数据核字（2008）第095132号

内 容 提 要

本书第1版于2007年被评为江苏省高等学校精品教材，第2版2007年入选普通高等教育“十一五”国家级规划教材。

本书保留经典数据结构知识，引入伸展树和跳表等新内容，教材反映抽象、封装和信息隐蔽等现代软件设计理念。本书重视程序设计和实践性。书中算法都有完整的C++程序，程序代码注释详细，结构清晰，构思精巧，它们既是很好的学习数据结构和算法的示例，也是很好的C++程序设计示例。实习指导和实习题独立成章，指导学生按软件工程学的方法设计算法，编写程序和书写文档。

本书可作为电气信息类、电子信息科学类、管理信息系统、电子商务、教育技术等相关专业数据结构课程的教材，也可供计算机软件及应用的工程技术人员参考。

普通高等教育“十一五”国家级规划教材
21世纪高等学校计算机规划教材
数据结构——使用C++语言描述（第2版）

◆ 主　　编　陈慧南
责任编辑　蒋　亮

◆ 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
北京铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：15.5
字数：403千字　　2008年10月第2版
印数：10 001 – 13 000册　　2008年10月北京第1次印刷
ISBN 978-7-115-18402-3/TP

定价：26.00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

计算机应用能力已经成为社会各行业最重要的工作要求之一，而计算机教材质量的好坏会直接影响人才素质的培养。目前，计算机教材出版市场百花争艳，品种急剧增多，要从林林总总的教材中挑选一本适合课程设置要求、满足教学实际需要的教材，难度越来越大。

人民邮电出版社作为一家以计算机、通信、电子信息类图书与教材出版为主的科技教育类出版社，在计算机教材领域已经出版了多套计算机系列教材。在各套系列教材中涌现出了一批被广大一线授课教师选用、深受广大师生好评的优秀教材。老师们希望我社能有更多的优秀教材集中地呈现在老师和读者面前，为此我社组织了这套“21世纪高等学校计算机规划教材——精品系列”。

“21世纪高等学校计算机规划教材——精品系列”具有下列特点。

（1）前期调研充分，适合实际教学需要。本套教材主要面向普通本科院校的学生编写，在内容深度、系统结构、案例选择、编写方法等方面进行了深入细致的调研，目的是在教材编写之前充分了解实际教学的需要。

（2）编写目标明确，读者对象针对性强。每一本教材在编写之前都明确了该教材的读者对象和适用范围，即明确面向的读者是计算机专业、非计算机理工类专业还是文科类专业的学生，尽量符合目前普通高等教学计算机课程的教学计划、教学大纲以及发展趋势。

（3）精选作者，保证质量。本套教材的作者，既有来自院校的一线授课老师，也有来自IT企业、科研机构等单位的资深技术人员。通过他们的合作使老师丰富的实际教学经验与技术人员丰富的实践工程经验相融合，为广大师生编写出适合目前教学实际需求、满足学校新时期人才培养模式的高质量教材。

（4）一纲多本，适应面宽。在本套教材中，我们根据目前教学的实际情况，做到“一纲多本”，即根据院校已学课程和后续课程的不同开设情况，为同一科目提供不同类型的教材。

（5）突出能力培养，适应人才市场要求。本套教材贴近市场对于计算机人才的能力要求，注重理论技术与实际应用的结合，注重实际操作和实践动手能力的培养，为学生快速适应企业实际需求做好准备。

（6）配套服务完善，共促提高。对于每一本教材，我们在教材出版的同时，都将提供完备的PPT课件，并根据需要提供书中的源程序代码、习题答案、教学大纲等内容，部分教材还将在作者的配合下，提供疑难解答、教学交流等服务。

在本套教材的策划组织过程中，我们获得了来自清华大学、北京大学、人民大学、浙江大学、吉林大学、武汉大学、哈尔滨工业大学、东南大学、四川大学、上海交通大学、西安交通大学、电子科技大学、西安电子科技大学、北京邮电大学、北京林业大学等院校老师的大力支持和帮助，同时获得了来自信息产业部电信研究院、联想、华为、中兴、同方、爱立信、摩托罗拉等企业和科研单位的领导和技术人员的积极配合。在此，人民邮电出版社向他们表示衷心的感谢。

我们相信，“21世纪高等学校计算机规划教材——精品系列”一定能够为我国高等院校计算机课程教学做出应有的贡献。同时，对于工作欠缺和不妥之处，欢迎老师和读者提出宝贵的意见和建议。

前　言

本书作者多年在南京邮电大学讲授“数据结构”和“算法设计与分析”课程，已出版《数据结构——C++语言描述》、《数据结构——C语言描述》、《数据结构与算法》、《算法设计与分析》等教材。本教材第1版于2007年被评为江苏省高等学校精品教材，此第2版于2007年入选普通高等教育“十一五”国家级规划教材。

本书按CC2001关于数据结构知识领域的要求，参照国外最新教材，结合教学实际编写而成。本书保留经典数据结构知识，引入伸展树和跳表等新内容，反映抽象、封装和信息隐蔽等现代软件设计理念，内容新旧取舍恰当，广度和深度适中。

本书对程序设计和实验赋予足够重视，将上机实习专门作为单独的一章编写，指导学生初步学习按软件工程学的方法设计和编写程序。书中算法都有完整的C++程序，程序代码注释详细，结构清晰，构思精巧，它们既是很好的学习数据结构和算法的示例，也是很好的C++程序设计示例。

本书结构严谨、内容深入浅出，配有大量的实例和图示，另有丰富的习题和实习题，易教易学。全书共分12章和附录。

第1章是基础知识，讨论有关数据结构、算法、数据抽象和抽象数据类型的概念，给出数据结构和算法的描述方法，讨论算法分析的基本方法。

第2～4章介绍几种线性数据结构：线性表、堆栈和队列、数组和字符串。第3章还讨论递归和递归算法，以及编译程序实现函数嵌套调用的机制。

第5章讨论树结构。第6～8章讨论集合的表示和运算。其中，第6章为顺序搜索和二分搜索，第7章为二叉搜索树、二叉平衡树、B树和伸展树，第8章为跳表和散列表。第9章讨论图。

第10章介绍多种内排序算法。第11章讨论外排序过程和算法。文件作为一种外存的数据结构，也在第11章介绍。

数据结构上机实习是课程教学不可或缺的重要环节，第12章和附录是专门为数据结构上机实习而编写的。第12章为实习指导和实习题。附录介绍调试技术和VC++调试器。

本书由陈慧南主编。其中，第1章、第5～9章和第11章由陈慧南编写，第2～4章和第10章由陈春玲编写，第12章由朱立华和陈慧南共同完成，附录由朱立华编写。全书由陈慧南统稿、审稿。

本书包含“数据结构”课程72学时的教学内容。对于学时数少于72学时的授课计划，可根据实际学时加以剪裁。作者已在目录中对难度较大，或非基本的章节标上*号，供读者选取时参考。除*号部分内容外的基本部分适合48～56学时授课计划。

本书在编写过程中得到了南京邮电大学和计算机学院领导的大力支持，在此表示衷心的感谢。

书中不当之处，敬请读者批评指正。

作　者

2008年8月1日于南京

目 录

第1章 基础知识

本章首先介绍什么是数据结构，算法和数据结构在计算机学科中的地位如何；然后介绍数据抽象和抽象数据类型概念，阐明其与数据结构的关系，给出数据结构和算法的描述方法；最后讨论算法分析的基本方法。

1.1 算法与数据结构

计算机由硬件和软件组成，硬件通过软件发挥效用。硬件是躯体，软件是灵魂，软件的核心是程序。学习程序设计需要掌握一门程序设计语言，它是学习计算机后续课程所必需的技能。但程序设计不等于编码，为了充分利用计算机资源，开发高效的程序，计算机人员还必须掌握计算机学科多方面知识，如数据的组织、算法的设计和分析、软件工程技术等。

随着计算机科学与技术的发展，计算机应用已远远超出了单纯进行科学计算的范围。从传统的应用领域，如工业控制、情报检索、企业管理、商务处理、图形图像、人工智能等诸多的数据处理领域，发展到电子政务、电子商务、办公自动化、企业资源管理、电子图书馆、远程教育、远程医疗等更广泛的领域。计算机技术已渗透到国民经济的各行各业和人们日常生活的方方面面。今天，信息技术作为现代技术的标志，已成为世界各国经济增长的主要动力。

现实世界各领域中的大量信息都必须转换成数据才能在计算机中存储、处理。数据是信息的载体，应用程序处理各种各样的数据。笼统地说，所谓**数据**，就是计算机加工处理的对象。数据一般分为两类：**数值数据**和**非数值数据**。数值数据是一些整数、实数或复数，主要用于工程计算、科学计算、商务处理等。非数值数据包括字符、文字、图形、图像、语音、表格等。这类数据的特点是量大，而且往往有着复杂的内在联系。如果单纯依靠改进程序设计技巧，已无法编制出高效可靠的程序，而必须对数据本身的结构加以研究。数据的组织和表示方法直接影响使用计算机求解问题的效率。算法设计通常建立在所处理数据的一定组织形式之上。在许多应用中，对于相同数据的同样处理要求，如果选择不同的数据结构，会有不同的处理效率：运算时间和存储空间。数据结构和算法两者是紧密结合的。

对计算机学科来说，数据结构与算法的概念是至关重要的，它们是计算机学科的基础之一，更是软件技术的基础。数据结构与算法之间有着本质的联系。当谈论一种算法时，自然要涉及算法所处理的数据问题；而讨论数据的组织或结构，离开了对处理此类数据的运算及其算法的研究也是无意义的。有人概括过一个公式，即

程序=算法+数据结构

在**计算学科教学计划1991**中，数据结构和算法是计算机学科的9个研究领域之一。**计算学科教学计划2001**调整成14个领域，数据结构和算法的基本内容主要涵盖在**算法与复杂性**和**程序设计基础**两个领域中。CC2001强调了算法和程序设计，它建议使用90个核心学时讲授这方面知识，占整个核心学时数的32.2%。

数据结构主要是为研究和解决如何使用计算机组织和处理非数值问题而产生的理论、技术和方法。它已成为计算机学科研究的基本课题之一。本书讨论数据的结构技术，以及在数据结构上的算法。

1.2 什么是数据结构

1.2.1 基本概念

前文提到，数据是计算机加工处理的对象。一个数据可以是由成分数据构成的具有某种结构的数据。在这里，我们称组成数据的成分数据为**数据元素**。通常，数据元素可以是简单类型的，如整数、实数、字符等，也可以是结构类型，如**记录**。如果把每个学生的记录看作一个数据元素，它包括学号、姓名、性别等**数据项**——数据项是不可再分割的，一个班学生的记录组成了表1-1所示的学生情况表。

表1-1　　学生情况表

学　号	姓　名	性　别	其他信息
B02040101	王小红	女	…
B02040102	林悦	女	…
B02040103	陈菁菁	女	…
B02040104	张可可	男	…
⋮	⋮	⋮	⋮

那么，什么是数据结构？数据结构是计算机科学与技术领域中广泛使用的术语，但究竟什么是数据结构，在计算机科学界至今没有标准的定义。

Sartaj Sahni在他的《数据结构、算法与应用》一书中称“数据结构是数据对象，以及存在于该对象的实例和组成实例的数据元素之间的各种**联系**。这些联系可以通过定义相关的函数来给出”。该书中，Sartaj Sahni将**数据对象**定义为“一个数据对象是**实例**或**值**的集合”。

Clifford A. Shaffer在《数据结构与算法分析》一书中的定义是：“数据结构是ADT[①]的物理实现。”

Lobert L.Kruse在他的《数据结构与程序设计》一书中，将一个数据结构的设计过程分成抽象层、数据结构层和实现层。其中，抽象层是指抽象数据类型层，它讨论数据的逻辑结构及其运算，数据结构层和实现层讨论一个数据结构的表示和在计算机内的存储细节以及运算的实现。

我们认为，一个**数据结构**是由数据元素依据某种逻辑联系组织起来的，对数据元素间逻辑关系的描述称为数据的逻辑结构；数据必须在计算机内存储，数据的存储结构是数据结构的实现形

① ADT（abstract data type，抽象数据类型）

式，是其在计算机内的表示；此外讨论一个数据结构必须同时讨论在该类数据上执行的运算才有意义。

本书中，我们将一个数据结构的讨论分为两个层次，即抽象层和实现层。抽象层讨论数据的逻辑结构及其运算定义，实现层讨论数据的存储表示以及运算的算法实现。

1.2.2　数据的逻辑结构

从概念上讲，一个数据结构是由数据元素依据某种逻辑联系组织起来的。对数据元素间逻辑关系的描述被称为数据的**逻辑结构**，它可以用一个二元组表示，即

$$DS = (D, R)$$

其中，D 是数据元素的有限集合，R 是 D 中元素序偶的集合。

例如，$DS=\{D,R\}$,$D=\{a,b,c,d\}$,$R=\{<a,b>,<b,c>,<c,d>\}$，其中，序偶$<a,b>$表示元素 a 和 b 之间的关系，我们称 a 是 b 的**直接前驱**，b 是 a 的**直接后继**。在不引起混淆的情况下，常简称直接前驱为“**前驱**”，直接后继为“**后继**”。数据结构 DS 是一个线性数据结构。

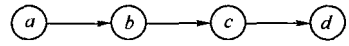


图 1-1　DS 示意图

我们用小圆圈表示数据元素，用带箭头的线表示元素间的关系。两个不同元素的序偶称为**边**，则 DS 可由图 1-1 表示。

根据数据结构中数据元素之间关系的不同特征，可划分为以下 4 种基本逻辑结构。

（1）集合结构

集合结构中，元素间的次序是随意的。元素之间除了“属于同一个集合”的联系之外没有其他关系。由于集合结构的元素间没有固有的关系，因此往往需要借助其他结构才能在计算机中实际表示此结构。

（2）线性结构

线性结构是数据元素的有序序列，其中，第一个元素没有前驱只有后继，最后一个元素只有前驱没有后继，其余元素有且仅有一个前驱和一个后继。

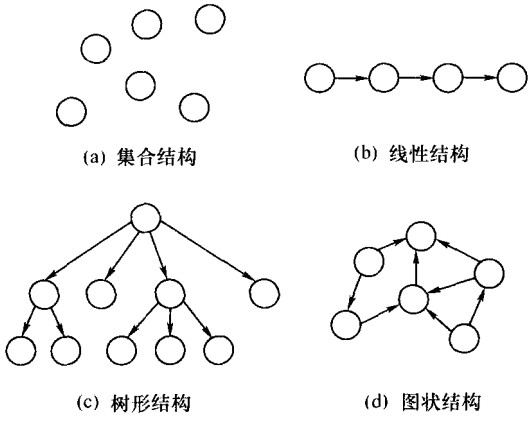


图 1-2　4 种基本的结构关系

（3）树形结构

树中，除一个特殊元素称为**根**，它没有前驱只有后继外，其余元素都有且仅有一个前驱，但后继的数目不限。对于非根元素，都存在着一条从根到该元素的**路径**。树是层次数据结构。

（4）图状结构

图是最一般的数据结构，图中每个元素的前驱和后继的数目都不限。

这 4 种基本结构还可进一步分成两类：**线性结构**和**非线性结构**。把除了线性结构以外的几种结构，即树、图和集合都归入非线性结构一类。图 1-2 所示为 4 种基本结构关系的示意图。

1.2.3　数据的存储表示

数据的逻辑结构是面向应用问题的，是从用户角度看到的数据结构。数据必须在计算机内存储，数据的**存储结构**是数据在计算机内的表示形式，是逻辑数据的存储映像，它是面向计算机的。

我们知道，计算机内存是由有限个存储单元组成的一个连续存储空间，这些存储单元是字节编址或者是字编址的。从存储器角度看，内存中是一堆二进制数据，它们可以被机器指令解释为指令、整数、字符、布尔数等，也可以被数据结构的算法解释为具有某种结构的数据。

对一个数据结构，找到一种有效的存储表示方法，使它适于计算机表示是十分重要的。**顺序**和**链接**是两种最基本的**存储表示**方法。

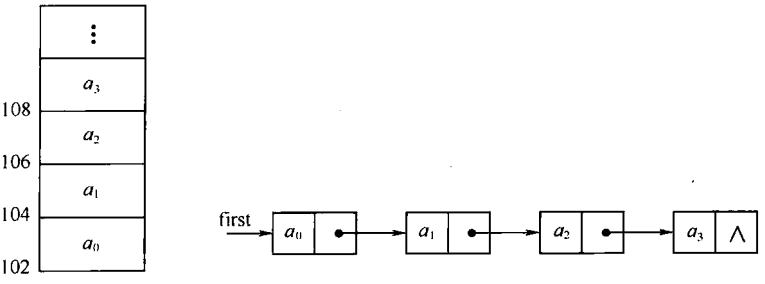
顺序（或称**连续**）表示方法需要一块连续的存储空间，并把逻辑上相关的数据元素依次存储在该连续的存储区中。例如，由 4 个元素组成的线性数据结构（a_0,a_1,a_2,a_3），存储在某个连续的存储区内，设存储区的起始地址是 102，假定每个元素占 2 个存储单元，则其顺序存储表示如图 1-3（a）所示。

在顺序表示下，可以容易地用一个数学公式来确定每个元素的存储地址。对于图 1-3（a）的顺序存储结构，可使用式（1-1）计算元素 a_k 的存储位置 $Loc(a_k)$，即

$$Loc(a_k)=102+2\times k \qquad (1\text{-}1)$$

顺序表示方法并不仅限于存储线性数据结构。对于非线性数据结构，如树结构，有时也可采用顺序存储的方法表示。这将在以后讨论。

另一种基本的存储表示方法是链接存储表示。在链接存储表示下，为了在计算机内存储一个元素，除了需要存放该元素本身的信息外，还需要存放与该元素相关的其他元素的地址信息。这两部分信息组成存放一个数据元素的**结点**。图 1-3（b）所示为线性结构（a_0,a_1,a_2,a_3）的链接存储表示。其中，每个结点的存储块分成两部分，一部分存放元素自身，另一部分包含该元素逻辑上的后继结点的存储地址。这种关于相关结点的地址信息被称为**链**。图 1-3（b）中，符号“∧”表示空链（即不代表任何具体结点的存储地址）。注意，一个结点的存储地址通常是指存放该结点的存储块的起始存储单元地址。



(a) 顺序存储结构　　(b) 链接存储结构

图 1-3　两种基本的存储表示方法

以后的讨论中，在不会引起混淆的场合，将不明确区分结点和元素这两个术语。但必要时，我们将包括地址信息在内的存储块整体称为**结点**，而将其中的元素信息部分称为该结点的**元素**。

除顺序和链接表示外，还可以有其他存储数据的方法，如**索引**方法和**散列**方法。关于这两种方法，请参看以后相关章节。

这些基本的存储表示方法，以及它们的组合，可衍生出各种数据的存储结构。

1.2.4　数据结构的运算

如果说数据的逻辑结构描述了数据的静态特性，那么在数据的逻辑结构上定义的一组运算给出了数据被使用的方式，即数据的动态特性。使用数据结构上定义的运算，用户可对数据结构的

实例或组成实例的数据元素实施相应的操作。运算的结果可使数据改变状态。

请注意，在下面的讨论中，我们一般不明确区分数据结构和它的实例。例如，我们称创建一个数据结构，很显然是指创建某种数据结构的一个实例。

数据结构的最常见的运算有：

（1）创建运算——创建一个数据结构；

（2）清除运算——删除数据结构中的全部元素；

（3）插入运算——在数据结构中插入一个新元素；

（4）删除运算——将数据结构中的某个指定元素删除；

（5）搜索运算——在数据结构中搜索满足一定条件的元素；

（6）更新运算——修改数据结构中某个指定元素的值；

（7）访问运算——访问数据结构中某个元素；

（8）遍历运算——按照某种次序，系统地访问数据结构的各元素，使得每个元素恰好被访问一次。

如果一个数据结构一旦创建，其结构不发生改变，则称为**静态数据结构**，否则称为**动态数据结构**。

堆栈是一种线性数据结构，可以向栈中加入元素，但只允许访问和删除最后入栈的元素。例 1.1 给出了在栈上定义的若干运算。

例 1.1　栈数据结构

（1）向栈中加入一个元素（Push 运算）；

（2）从栈中删除最后加入的元素（Pop 运算）；

（3）访问最后加入栈中的元素（Top 运算）。

堆栈数据结构还可以包括更多的运算，有关堆栈的详细内容将在第 3 章讨论。

研究数据结构是为了解决应用问题，研究一个数据结构，必须同时研究在该数据结构上执行的相关运算及其算法才有意义。数据和操纵数据的运算是研究数据结构不可分割的两个方面。所以，在讨论数据结构时，不但要讨论数据的逻辑结构和存储结构，还要讨论在数据结构上执行的运算，以及实现这些运算的算法。

通过对运算及其算法的性能分析和讨论，使得我们在求解应用问题时，能有的放矢地选择和设计适当的数据结构，编写出高效的程序。

1.3　数据抽象和抽象数据类型

随着软件规模的不断扩大，软件开发难度越来越大。抽象、封装和信息隐蔽是控制软件开发复杂度、提高软件可靠性的重要手段。使用数据抽象可以降低程序设计的复杂度，提高可靠性，同时也提高了软件开发的效率。本书采用抽象数据类型的观点讨论数据结构。一个数据结构被作为一个抽象数据类型来描述和实现。

1.3.1　抽象、数据抽象和过程抽象

抽象可以理解为一种机制，其实质是抽取共同的和本质的内容，忽略非本质的细节。抽象可以使一个问题的求解过程采用自顶向下的方式分步进行：首先考虑问题的最主要方面，然后再逐

步细化，进一步考虑问题的细节，并最终实现之。

在程序设计中，抽象机制被用于两个方面：数据和过程。**数据抽象**使程序设计者可以将数据元素间的逻辑关系和数据在计算机内的具体表示分别考虑。而**过程抽象**可使程序设计者将一个运算的定义与实现运算的具体方法分开考虑。抽象的好处主要在于降低了问题求解的难度。

1.3.2 封装与信息隐蔽

封装是指把数据和操纵数据的运算组合在一起的机制。使用者只能通过一组允许的运算访问其中的数据。原则上，数据的使用者只需知道这些运算的定义（也称规范）便可访问数据，而无需了解数据是如何组织和存储的以及运算算法的实现细节，也就是说对使用者隐藏了数据结构以及程序的实现细节。这种设计数据结构或程序的策略称为**信息隐蔽**。

可以将数据和操纵数据的运算组成**模块**。每个模块有一个明确定义的**界面**，模块内部信息只能经过这一界面被外部访问。一个模块的界面是实现运算的一组**函数**。模块应采用封装和信息隐蔽原则来设计，这样的模块被称为**黑盒子**。一个模块可以被其他程序或模块调用，它们是该模块的**客户**。

封装和信息隐蔽是把错误局部化的有效措施，有助于降低问题求解的复杂性，也提高了程序的可靠性。

1.3.3 数据类型和抽象数据类型

读者已熟悉C++语言的基本数据类型，它们包括字符型、整型、实型、指针类型等原子类型。原子数据类型的值是不可分割的。现实世界最终可以用这些初等数据来表示。另一类是结构类型。结构类型数据的值是由若干成分按某种结构组成的，因此是可以分解的。

数据类型是程序设计语言中的概念，它是数据抽象的一种方式。一个**数据类型**定义了一个值的集合以及作用于该**值集**的运算集合。程序设计语言中，一个数据类型不仅规定了该类型的变量（或常量）的取值范围，还定义了该类型允许的运算。

例如，一个整型类型（int）变量的取值范围是-32768～32767，在整型数上允许的运算包括算术运算（+、-、*、/、%）、关系运算（<、>、<=、>=、==、!=）、赋值运算（=）等。图1-4所示的第一部分是类型名称，第二部分是该类型的规范，它规定了整型变量（或常量）的取值范围及允许的运算，第三部分是与该类型实现有关的方面，包括数据的表示和运算的实现。可见，一个数据类型的规范和实现是分离的。

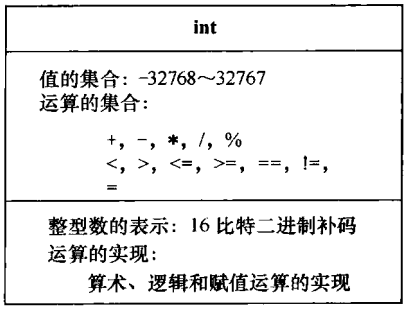


图1-4 C/C++的整型类型（int）

了解一个数据类型的对象（变量、常量）在计算机内的表示是有用的，但往往也是危险的。这使得应用程序可以直接编写算法操纵该类型对象，而不是严格使用预先定义的运算进行访问，这会滋生不可预知的错误。而且一旦改变该类型的存储表示，则必须改变所有使用该类型对象的应用程序。目前普遍认为对使用者隐藏一个数据类型的存储表示是一个好的设计策略，用户程序只能使用该类型提供的运算（函数）访问该类型对象，这就是抽象数据类型。

一个**抽象数据类型（abstract data type，ADT）**是一个数据类型，其主要特征是该类型的对象及其运算的规范，与该类型对象的表示和运算的实现分离，实行封装和信息隐蔽，即所谓使用

和实现分离。

那么，在一个抽象数据类型上应当定义哪些运算？这取决于应用。如果我们使用一个抽象数据类型上定义的运算，足以对该类型对象实施所需要的全部操作，则这组运算被认为是完备的。

其实，C++语言的整型 int 就是一个抽象数据类型，我们只能通过整型 int 所规定的运算操纵整型变量或常量。但 C++的数组和结构不是抽象数据类型，我们既不能把一个数组整体赋值给另一个数组，数组也不能作为函数值返回。结构虽然可以赋值，但与数组一样不能直接将输入输出符用于一个结构。

对于用户自定义的数据类型，只有当面向对象程序设计语言出现后，才提供了必要的机制，使用户可以真正定义抽象数据类型，并实现之。例如，可使用 C++语言的**类**的封装和信息隐蔽机制来实现抽象数据类型。

1.3.4　数据结构与抽象数据类型

对于一个数据结构，一方面要声明它的逻辑结构，同时还应定义一组在该数据结构上执行的运算。逻辑结构和运算的定义组成了数据结构的**规范**，而数据的存储表示和运算算法的描述构成数据结构的**实现**。规范是实现的准则和依据，规范指明“做什么”，而实现解决“怎样做”。数据结构的用户（即**客户**）通过规范中描述的界面使用一个数据结构，而不必了解实现细节。从规范和实现两方面来讨论数据结构的方式是抽象数据类型的观点。这里一种数据结构被视为一个抽象数据类型。

1.4　描述数据结构和算法

前面的讨论认为，一个数据结构包含两个抽象层次，即数据结构的规范和数据结构的实现。数据结构的规范是指它的逻辑结构和运算的规范，数据结构的实现是指它的存储表示和运算的实现。

1.4.1　数据结构的规范

在概念层次上，数据的逻辑结构分为四类，即集合结构、线性结构、树形结构和图状结构。其中，每一种结构又可进一步细分，得到多种数据结构。有时一种数据结构需借助其他数据结构来表示，比如集合结构可以使用线性表、搜索树或散列表来表示。

一个运算的规范是在概念的层次上，或从用户角度对运算的精确定义，它应能精确描述运算而又不涉及运算的实现细节。本书中，我们使用如 ADT 1.1 所示的格式化语言和程序 1.1 的 C++模板抽象类两种形式来共同描述例 1.1 的栈数据结构的规范。

ADT 1.1　栈 ADT

```
ADT Stack {
```

数据：

0 个或多个元素的线性序列（$a_0, a_1, \ldots, a_{n-1}$），其最大允许长度为 MaxStackSize。

运算：

Create()：建立一个空栈。

Destroy()：撤销一个栈。

Push(x)：值为 x 的新元素进栈，成为栈顶元素。

Pop()：从栈中删除栈顶元素。

Top(x)：在 x 中返回栈顶元素。

}

ADT 1.1 采用格式化自然语言来描述栈，这是一种非形式描述。堆栈被定义为元素的线性序列（$a_0,a_1,...,a_{n-1}$），在栈上定义了一组运算，如 Push，Pop，Top 等。为了更严格地规定栈的用户函数界面，可定义一个抽象模板类。程序 1.1 将栈定义为一个抽象模板类 Stack。栈的所有实现都将作为该类的派生类，都必须重定义类 Stack 的成员函数，从而保证了实现栈 ADT 的各个派生类都有完全一致的用户界面。

程序 1.1　栈的模板抽象类

```
template<class T>
class Stack
{// 栈类 Stack 是一个模板抽象类，其成员函数为纯虚函数，未定义数据成员。
public:
   virtual bool Push(T x)=0;
   virtual bool Pop()=0;
   virtual bool Top(T &x) const=0;
    ⋮
};
```

1.4.2　实现数据结构

实现一个数据结构必须首先确定数据的存储表示，然后在给定的存储方式下实现相应的运算。本书使用 C++语言的数据类型（如结构、数组、指针等）描述数据的存储表示，并使用 C++语句描述实现运算的算法。这里，我们不妨假定 C++语言的数组和结构都是顺序存储的：数组元素具有相同的数据类型，按照下标的次序存储在连续的存储区；结构可以由不同类型的域（成分）组成，按照域表的次序顺序存放。借助于 C++语言的数组、结构和指针使我们可以描述和实现数据结构的各种不同的存储表示。

程序 1.2 的类 SeqStack 是 ADT 1.1 的顺序存储表示下的实现，它采用一维数组存储堆栈元素，它是 Stack 类的派生类，其中每个成员函数实现一个栈运算。

程序 1.2　栈的顺序实现

```
template<class T>
class SeqStack:public Stack<T>
{
public:
    ⋮
private:
    int top;                    //top 记录最后入栈的元素在一维数组 s 的下标
    int maxTop;                 //最大栈顶指针
    T *s;                       //s 指向动态生成的一维数组，存放栈中元素
};
template<class T>
SeqStack<T>::SeqStack(int mSize)
{
    maxTop=mSize-1;             //设置栈的容量值
    s=new T[mSize];             //生成存储栈的数组
```