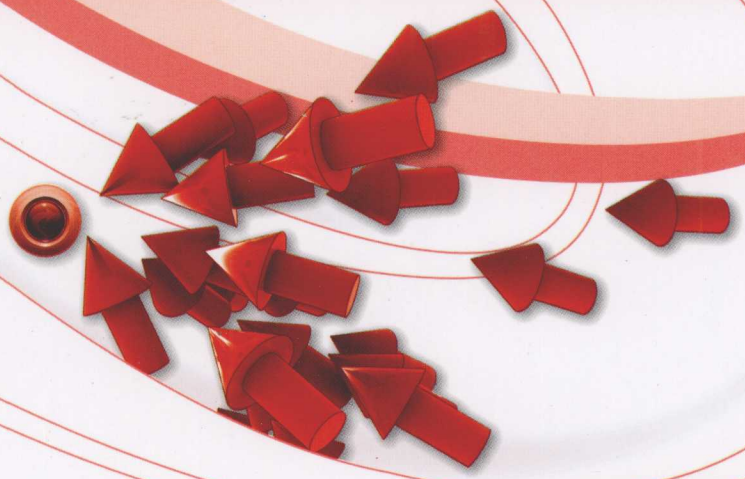


高等学校计算机规划教材

数据结构（C++版）（第2版）

叶核亚 编著

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

高等学校计算机规划教材

数据结构（C++版）

（第2版）

叶核亚　编著

電子工業出版社

Publishing House of Electronics Industry

北京・BEIJING

内 容 简 介

本书全面系统地介绍数据结构的基础理论和算法设计方法，包括线性表、树、图等数据结构以及查找和排序算法。内容涉及的广度和深度符合计算机专业本科的基本要求，体现了本科教学的培养目标。

本书采用 C++语言，以面向对象方法描述数据结构和算法。本书理论叙述精练，结构安排合理，重点是数据结构设计和算法设计，通过降低理论难度和抽象性、加强实践环节等措施，力求增强学生的理解能力和应用能力。

本书有配套的教学资料包，包括电子课件、源代码及习题解答等。

本书可作为普通高等学校计算机及相近专业本科数据结构课程的教材，也可作为从事计算机软件开发和工程应用人员的参考书。

图书在版编目(CIP)数据

数据结构：C++版/叶核亚编著. —2 版. —北京：电子工业出版社，2009.1
高等学校计算机规划教材
ISBN 978-7-121-08004-3

Ⅰ. 数… Ⅱ. 叶… Ⅲ. ① 数据结构－高等学校－教材 ② C++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 199775 号

策划编辑：章海涛
责任编辑：章海涛
印　　刷：
装　　订：北京市李史山胶印厂
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：20.25　字数：512 千字
印　　次：2009 年 1 月第 1 次印刷
印　　数：5 000 册　　定价：29.00 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。

第 2 版前言

数据结构是软件设计的重要理论和实践基础，数据结构设计和算法设计是软件系统设计的核心。“数据结构”课程讨论的知识内容是软件设计的理论基础，“数据结构”课程介绍的技术方法是软件设计中使用的基本方法。“数据结构”是理论与实践并重的课程，要求学生既要掌握数据结构的基础理论知识，又要掌握运行和调试程序的基本技能。因此，“数据结构”课程在计算机类各专业本科学生的培养过程中有着十分重要的地位，是计算机类专业的一门核心课程，是培养程序设计能力的必不可少的重要环节。

“数据结构”课程内容多，概念抽象，理论深奥，递归算法难度较大，一直是计算机专业最难学的课程之一。**本书精选基础理论内容，重点是数据结构设计和算法设计，通过降低理论难度和抽象性、加强实践环节等措施，进一步增强学生的理解能力和应用能力，力求取得较好的教学效果。**

本书的特色说明如下。

（1）内容全面、注重基础

本书全面、系统地介绍数据结构的基础理论和算法设计方法，阐明线性表、树、图等数据模型的逻辑结构，讨论它们在计算机中的存储结构，讨论每种数据结构所能进行的多种操作，以及这些操作的算法设计与实现；针对软件设计中应用频繁的查找和排序问题，根据不同数据结构对操作的实际需求，给出多种查找和排序算法，并分析算法的执行效率。

本书内容选择适合工科院校，理论叙述精练，简明扼要，结构安排合理，由浅入深，层次分明，重点突出，算法分析透彻，程序结构严谨规范。内容涉及的广度和深度符合本科培养目标的要求。

（2）采用 C++语言和面向对象程序设计思想描述数据结构和算法

数据结构和算法的设计不依赖于程序设计语言，但数据结构和算法的实现依赖于程序设计语言。描述数据结构所采用的软件方法和算法语言，需要随着软件方法及算法语言的不断发展而发展。面向对象程序设计方法是软件分析、设计和实现的新方法，是目前软件开发的主流方法。

C++语言是目前功能最强、应用广泛的支持面向对象程序设计的代表语言。C++语言具备表达数据结构和算法的基本要素。因此，采用 C++语言描述数据结构和算法不仅可行，也是“数据结构”课程内容教学改革的必然，完全符合本科培养目标的要求。**本书采用 C++语言描述数据结构和算法，算法以函数方式呈现，函数有明确的输入参数和返回值，以面向对象程序设计思想贯穿始终，使“数据结构”课程真正成为有效培养程序设计能力的训练课程，并与程序设计语言课程更好地衔接。**

本书只是借用 C++语言作为数据结构和算法的描述语言，**重点仍是数据结构和算法设计本身，而不是表现 C++语言的复杂技术**。因此，使用 C++语言的基本原则是，尽可能使用 C++语言的最基本成分（如数据类型、数组、函数、指针、类等）描述数据结构和表达算法设计思想，展现原始设计能力，使程序简洁、明了，算法思路清楚、明白，淡化或回避 C++

语言的友元类、运算符重载、多重继承等技术。

当一个问题有多种解决办法时，尽可能采用简单、直接并且不造成歧义的办法。例如，对数组元素进行操作，尽量采用下标形式，避免使用指针形式；函数的返回结果尽量采用返回值形式，避免使用指针的指针；构造函数尽量采用重载方式，避免采用默认值而造成的歧义等。

（3）增强实际应用

数据结构是一门理论和实践紧密结合的课程，因此需要在透彻理解理论知识的基础上，通过实践性环节，逐步锻炼学生的程序设计能力。

注重传授基础理论知识，注重在实践环节中培养程序设计的基本技能，是本书第 1 版的重要特色，也是本书的重要特色。本书精心选择并设计一系列与实际应用息息相关的例题、习题、实验题、课程设计题等，使原本枯涩难懂的理论变得生动有趣，并以此说明数据结构和算法的必要性，以及它们对实际应用程序设计的指导作用。同时，要求学生能在生活中发现问题并解决问题，提高学习兴趣，力求在潜移默化中使学生理解和体会理论知识的重要性，并掌握如何使用它们的方法。

除了每章的实验题给出详细的训练目标、设计内容和设计要求之外，针对课程设计实践性环节，本书给出了多种算法设计与分析的综合应用程序设计实例，详细说明需求方案、设计思想、模块划分、功能实现、调试运行等环节的设计方法，贯彻理论讲授与案例教学相结合的教学方法。

程序设计有其自身的规律，不是一蹴而就的，也没有捷径。程序员必须具备基本素质，必须掌握程序设计语言的基本语法以及算法设计思想和方法，并且需要积累许多经验。这个逐步积累的过程需要一段时间，需要耐心，厚积而薄发。

本书第 1～9 章是数据结构课程的主要内容，包括线性表、树、图等数据结构设计以及查找和排序算法设计；第 10 章为综合应用设计，包括课程设计范例和参考选题；第 11 章介绍如何使用 Visual C++ 6.0 集成开发环境，重点和难点是程序调试技术。第 11 章虽然放在最后，却是贯穿始终的，配合各章逐步进行。熟悉集成开发环境、掌握程序调试技术也需要经过一个逐步积累的过程。

本书所有程序均已在 Visual C++ 6.0 开发环境中调试通过。

本书由叶核亚编著，在写作过程中作者得到了许多帮助和支持，感谢南京大学计算机科学与技术系陈本林教授认真、细致地审阅了全稿，提出了许多宝贵意见；感谢陈立、廖雷、阚建飞、陈建红、陈瑞、王青云、李林广、刁翔、沈晨鸣、王少东等老师给予的帮助；感谢电子工业出版社的支持；感谢众多读者朋友对本书第 1 版提出的宝贵意见。

对书中存在的不妥与错漏之处，敬请读者朋友批评指正。

本书为任课教师提供配套的教学资源（包含电子课件和例题源代码、习题解答），需要者可**登录到华信教育资源网站**（http://www.huaxin.edu.cn 或 http://www.hxedu.com.cn），注册之后进行**下载**，或发邮件到 unicode@phei.com.cn **咨询**。

作　者

目　录

第1章 绪 论

计算机数据处理的前提是数据组织，如何有效地组织数据和处理数据是软件设计的基本内容，也是“数据结构”课程的基本内容。

作为绪论，本章勾勒数据结构课程的一个轮廓，说明数据结构课程的目的、任务和主要内容。本章主要介绍数据结构概念所包含的数据逻辑结构、数据存储结构和数据操作等，介绍抽象数据类型概念，介绍算法概念、算法设计目标、算法描述和算法分析方法。

1.1 数据结构的基本概念

1.1.1 为什么要学习数据结构

软件设计是计算机学科的核心内容之一。进行软件设计时要考虑的首要问题是数据的表示、组织和处理方法，这直接关系到软件的工程化程度和软件的运行效率。

随着计算机技术的飞速发展，计算机应用从早期的科学计算扩大到过程控制、管理和数据处理等领域。计算机处理的对象也从简单的数值数据，发展到各种多媒体数据。软件系统处理的数据量越来越大，数据的结构也越来越复杂。因此，针对实际问题，如何合理地组织数据，如何建立合适的数据结构，如何设计好的算法，是软件设计的重要问题，而这些正是“数据结构”课程讨论的主要内容。

在计算机中，现实世界中的对象用数据来描述。“数据结构”课程的任务是，讨论数据的各种逻辑结构、在计算机中的存储结构以及各种操作的算法设计。数据结构课程的主要目的是，培养学生掌握处理数据和编写高效率软件的基本方法，为学习后续专业课程以及进行软件开发打下坚实基础。

数据结构是软件设计的重要理论和实践基础，数据结构设计和算法设计是软件系统设计的基础和核心。“数据结构”课程讨论的知识内容是软件设计的理论基础，“数据结构”课程介绍的技术方法是软件设计中使用的基本方法。“数据结构”是一门理论与实践并重的课程，学生既要掌握数据结构的基础理论知识，又要掌握运行和调试程序的基本技能。因此，“数据结构”课程在计算机学科本科培养过程中的地位十分重要，是计算机专业本科的核心课程，是培养程序设计能力的必不可少的重要环节。

在计算机界流传着一句经典名言“数据结构+算法=程序设计”（瑞士 Niklaus Wirth 教授），这句话简洁、明了地说明了程序（或软件）与数据结构和算法的关系，以及“数据结构”课程的重要性。

1.1.2 什么是数据结构

数据（data）是描述客观事物的数字、字符以及所有能输入到计算机中并能被计算机接受的各种符号集合的统称。数据是信息的符号表示，是计算机程序的处理对象。除了数值数据，计算机能够处理的数据还可以是各种非数值数据，如字符串、图形、图像、音频、视频

等多媒体数据。

表示一个事物的一组数据称为一个**数据元素**（data element），数据元素是数据的基本单位。一个数据元素可以是一个不可分割的原子项，也可以由多个数据项组成。**数据项**（data item）是数据元素中有独立含义的、不可分割的最小标识单位。例如，一个整数、一个字符都是原子项，一个学生数据元素由学号、姓名、性别和出生日期等多个数据项组成。**关键字**（key）是数据元素中用于识别该元素的一个或多个数据项，能够唯一识别数据元素的关键字称为**主关键字**（primary key）。

在由数据元素组成的数据集合中，数据元素之间通常具有某些内在联系。研究数据元素之间存在的关系并建立数学模型，是设计有效地组织数据和处理数据方案的前提。

数据的结构是指数据元素之间存在的关系。一个**数据结构**（data structure）是由 n（$n \geqslant 0$）个数据元素组成的有限集合，数据元素之间具有某种特定的关系。

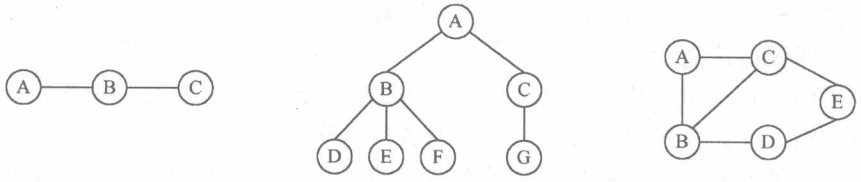
数据结构概念包含三方面：数据的逻辑结构、数据的存储结构和数据的操作。

1. 数据的逻辑结构

数据的逻辑结构是指数据元素之间的逻辑关系，用一个数据元素的集合和定义在此集合上的若干关系来表示，常被简称为数据结构。

根据数据元素之间逻辑关系的不同数学特性，数据结构可分为三种：线性结构、树结构和图结构（如图 1.1 所示），其中树和图又称为非线性结构。

图 1.1 以图示法表示了数据的逻辑结构，一个圆表示一个数据元素，圆中的字符表示数据元素的标记或取值，连线表示数据元素之间的关系。



（a）线性结构　　（b）树结构　　（c）图结构

图 1.1　三种数据结构

（1）线性结构

线性结构是最简单的数据结构，数据元素之间具有线性关系，即除第一个和最后一个元素外，每个元素有且仅有一个直接前驱元素和一个直接后继元素，第一个元素没有前驱元素，最后一个元素没有后继元素。在图 1.1（a）中，元素 B 的前驱是 A，后继是 C，元素 A 没有前驱，元素 C 没有后继。线性表、串、栈和队列等都是线性结构。

数据元素可以是数字、字符、字符串或其他复杂形式的数据，如整数序列{1, 2, 3, 4, 5, 6}，字母序列{'A', 'B', 'C',…, 'Z'}，表 1-1 所示的学生序列也是线性表，数据元素之间具有顺序关系。学生数据元素由“学号”、“姓名”、“年龄”等多个数据项组成，“姓名”可以作为标识一个学生的关键字，但不是主关键字；“学号”是能够唯一标识一个学生的主关键字。

表 1-1　学生信息表

学　号	姓　名	年　龄	学　号	姓　名	年　龄
20020001	王红	18	20020003	吴宁	18
20020002	张明	19	20020004	秦风	17

（2）树结构

树结构是数据元素之间具有层次关系的一种非线性结构，树中的数据元素通常称为结点。树结构的层次关系是指，根结点（最顶层）没有前驱结点（称为父母结点），除根之外的其他结点有且仅有一个父母结点，所有结点可有零到多个直接后继结点（称为孩子结点）。在图 1.1（b）中，A 是树的根结点，B 结点有一个父母结点 A，且有 3 个孩子结点 D、E 和 F。家谱、Windows 文件系统的组织方式、淘汰赛的比赛规则等都是树结构。具有树结构的淘汰赛比赛规则如图 1.2 所示，其中的数据是 2006 年世界杯足球赛淘汰赛的比赛结果。

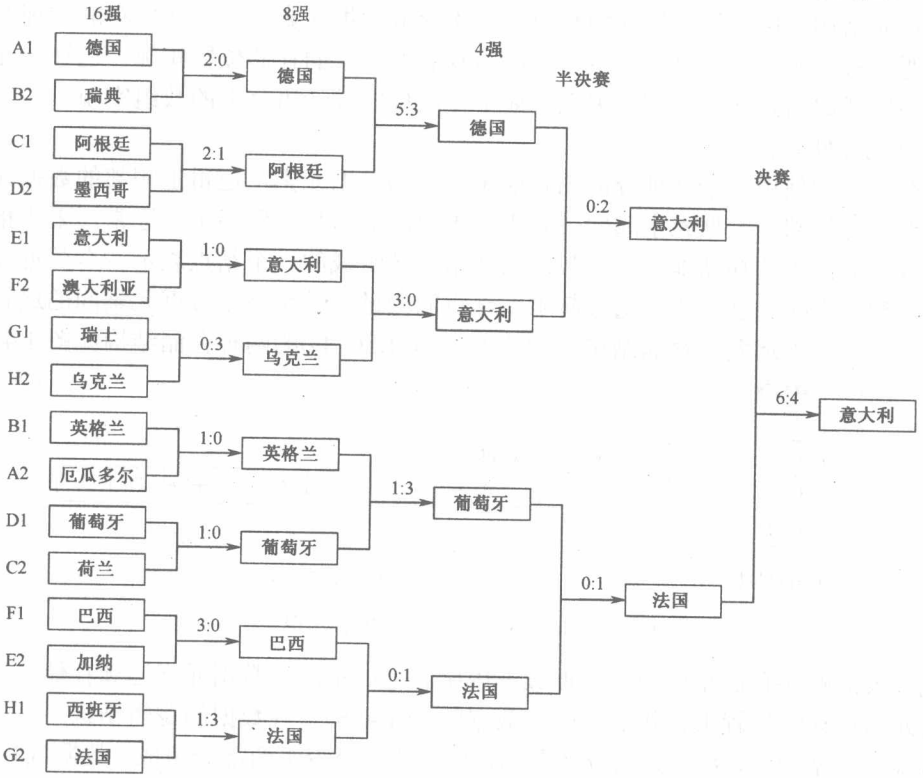


图 1.2　具有树结构的淘汰赛比赛规则

（3）图结构

图也是非线性结构，每个数据元素可有多个直接前驱元素和多个直接后继元素。例如，交通道路图、飞机航班路线图等都具有图结构。图 1.3 是从南京飞往昆明的航班路线图，有直飞航班，也有经停重庆或长沙的航班，连线边上数值表示两地间的千米数。

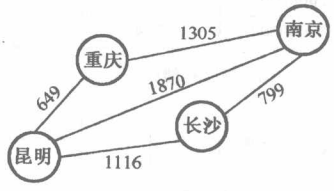


图 1.3　南京飞往昆明的航班路线图

2. 数据的存储结构

数据元素及其关系在计算机中的存储表示或实现称为数据的存储结构，也称为物理结构。软件系统不仅要存储所有数据，还要正确地表示出数据元素之间的逻辑关系。

数据的逻辑结构是从逻辑关系角度观察数据，与数据的存储无关，是独立于计算机的。而数据的存储结构是逻辑结构在计算机物理存储中的实现，是依赖于计算机的。

数据存储结构的基本形式有两种：顺序存储结构和链式存储结构。

（1）顺序存储结构

顺序存储结构使用一组连续的内存单元依次存放数据元素，元素在内存的物理存储次序与它们的逻辑次序相同，即每个元素与其前驱及后继元素的存储位置相邻。这样，元素的物理存储次序体现数据元素间的逻辑关系，通常使用程序设计语言中的数组实现。

（2）链式存储结构

链式存储结构使用若干地址分散的存储单元存储数据元素，逻辑上相邻的数据元素在物理位置上不一定相邻，数据元素间的关系需要采用附加信息特别指定。通常，采用指针变量记载前驱或后继元素的存储地址，由数据域和指针域组成的一个结点表示一个数据元素，通过指针域把相互直接关联的结点链接起来，结点间的链接关系体现数据元素间的逻辑关系。

线性表可采用上述两种存储结构。如线性表(A, B, C, D)的两种存储结构如图 1.4 所示。

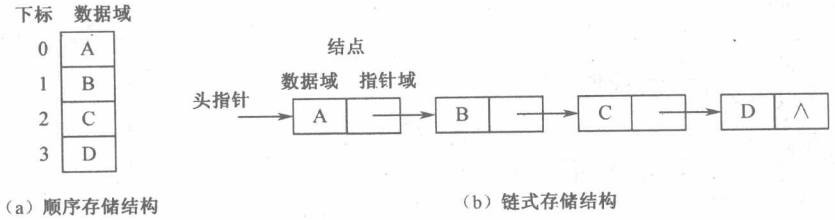


图 1.4 线性表(A, B, C, D)的两种存储结构

在线性表的顺序存储结构中，数据域占用所有存储空间，数据元素连续存储，逻辑上相邻的数据元素在存储位置上也相邻，因此数据的存储结构体现数据的逻辑结构。

在链式存储结构中，数据元素分散存储，每个结点至少由两部分组成：数据域和指针域，分别保存数据元素及前驱和（或）后继结点的地址。结点间的链接关系体现数据的逻辑结构。

如果一个数据元素由多个数据项组成，则数据域有多个。例如，学生信息表的顺序存储结构和链式存储结构如图 1.5 所示。

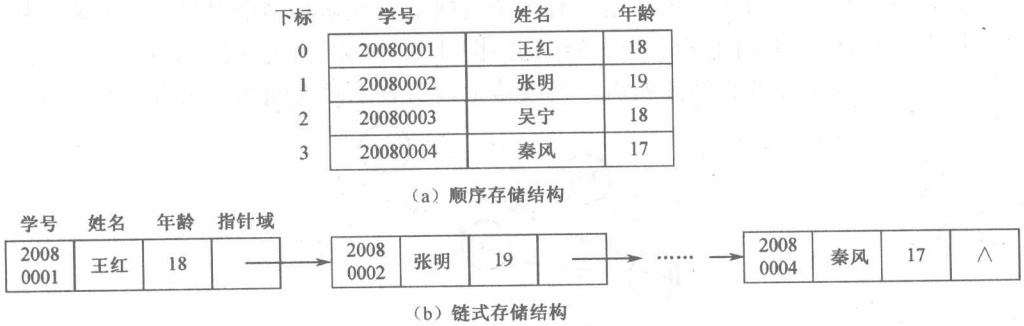


图 1.5 学生信息表的两种存储结构

顺序存储结构和链式存储结构是两种最基本、最常用的存储结构。除此之外，将顺序存储结构和链式存储结构进行组合，还可以构造出更复杂的存储结构。

3. 数据操作

数据操作是指对一种数据结构中的数据元素进行各种运算或处理。每种数据结构都有一组数据操作，其中包含以下基本操作。

- ⊙ 初始化。
- ⊙ 判断是否空状态。
- ⊙ 求长度：统计元素个数。
- ⊙ 包含：判断是否包含指定元素。
- ⊙ 遍历：按某种次序访问所有元素，每个元素只被访问一次。
- ⊙ 取值：获取指定元素的值。
- ⊙ 置值：设置指定元素的值。
- ⊙ 插入：增加指定元素。
- ⊙ 删除：移去指定元素。

数据操作定义在数据的逻辑结构上，对数据操作的实现依赖于数据的存储结构。例如，线性表包含上述一组数据操作，采用顺序存储结构或链式存储结构，都可实现这些操作。

1.1.3 数据类型与抽象数据类型

1. 数据类型

类型（type）是具有相同逻辑意义的一组值的集合。**数据类型**（data type）是指一个类型和定义在这个类型上的操作集合。数据类型定义了数据的性质、取值范围以及对数据所能进行的各种操作。例如，C++语言的整数类型 int，除了数值集合$[-2^{31}, \cdots, -2, -1, 0, 1, 2, \cdots, 2^{31}-1]$之外，还包括在这个值集上的操作集合[+, −, *, /, %, =]。

程序中的每个数据都属于一种数据类型，决定了数据的类型也就决定了数据的性质以及对数据进行的运算和操作，同时数据也受到类型的保护，确保对数据不能进行非法操作。

高级程序设计语言通常预定义一些基本数据类型和构造数据类型。基本数据类型的值是单个的、不可分解的，它可直接参与该类型所允许的运算。构造数据类型是使用已有的基本数据类型和已定义的构造数据类型按照一定的语法规则组织起来的较复杂的数据类型。构造数据类型的值由若干元素组合而成，这些元素按某种结构组织在一起。

C++语言的基本数据类型有整数类型、浮点数类型、字符类型等，构造数据类型有数组、结构体和文件等。例如，学生数据元素可声明为如下结构体类型，也可声明为类（class）。

```
struct Student
{
    char number[10];                    //学号
    char name[20];                      //姓名
    int age;                            //年龄
};
```

由若干学生组成的数据序列可声明存储在一个数组（array）中。例如：

```
Student group[50];
```

数据类型与数据结构这两个概念的侧重点不同。数据类型研究的是每种数据所具有的特性，以及对这种特性的数据能够进行哪些操作；数据结构研究的是数据元素之间具有的相互关系，与数据元素的数据类型无关，也不随数据元素值的变化而改变。

2. 抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是指一个数学模型以及定义在该模型上的一组操作。例如，复数是数学中常用的一种类型，但多数程序设计语言没有提供复数类型。一个复数 $a+bi$ 由实部 a 和虚部 b 两部分组成，i 是虚部标记。复数抽象数据类型描述如下：

```
ADT Complex                                    //复数抽象数据类型
{
    double real, im;                           //复数的实部和虚部
    Complex(real, im);                         //指定实部和虚部构造一个复数
    Complex add(Complex c);                    //加法，返回当前复数与 c 相加之后的复数
    Complex sub(Complex c);                    //减法，返回当前复数与 c 相减之后的复数
};
```

抽象数据类型和数据类型本质上是一个概念，“抽象”的含义是“定义与实现相分离”，即将一个类型上的数据及操作的逻辑含义与具体实现分离。程序设计语言提供的数据类型是抽象的，仅描述数据的特性和对数据操作的语法规则，并没有说明这些数据类型是如何实现的。程序员按照语言规则使用数据类型，只考虑对数据执行什么操作（做什么），而不必考虑怎样实现这些操作（怎样做）。程序设计语言实现了它预定义数据类型的各种操作。例如，赋值语句的语法定义如下：

```
变量=表达式
```

它表示先求得指定表达式的值，再将该值赋给指定变量。程序员需要关注所用数据类型的值能够参加哪些运算、表达式是否合法、表达式类型与变量类型是否赋值相容等；至于如何存储一个整数、变量的存储地址是什么、如何求得表达式值等实现细节则不必关注，这些操作由语言的实现系统完成。

ADT 的规范描述包括 ADT 名称、数据描述和操作描述，操作描述包括操作名、初始条件和操作结果。例如，集合 ADT 描述如下：

```
ADT Set
{
  数据：集合中有 n（n≥0）个数据元素，元素类型为 T
  操作：
    bool isEmpty();                            //判断集合是否为空
    int length();                              //返回集合的元素个数
    bool contain(T x);                         //判断集合是否包含指定元素 x
    bool add(T x);                             //增加指定元素 x
    bool remove(T x);                          //移去首次出现的指定元素 x
    void clear();                              //清空集合元素
    void print();                              //输出集合中所有元素
    bool equals(Set s);                        //比较当前集合与集合 s 是否相等
    bool containAll(Set s);                    //判断当前集合是否包含集合 s 中的所有元素
    bool addAll(Set s);                        //增加集合 s 中的所有元素，集合并
    bool removeAll(Set s);                     //移去那些也包含在集合 s 中的元素，集合差
    bool retainAll(Set s);                     //仅保留那些也包含在集合 s 中的元素
}
```

类似地，可将线性表、栈、队列、串、树、二叉树、图等数据结构分别定义为抽象数据类型，描述相应数据结构的逻辑特性和操作，与该数据结构在计算机内的存储及实现无关。

抽象是研究复杂对象的基本方法，也是一种信息隐蔽技术，从复杂对象中抽象出本质特征，忽略次要细节，使实现细节相对于使用者不可见。抽象层次越高，其软件复用程度也越高。抽象数据类型是实现软件模块化设计思想的重要手段。一个抽象数据类型是描述一种特定功能的基本模块，由各种基本模块可组织和构造起来一个大型软件系统。

1.2　算法

1.2.1　什么是算法

1. 算法定义

曾获图灵奖的著名计算科学家 D.knuth 对算法做过一个为学术界广泛接受的描述性的定义。一个**算法**（algorithm）是一个有穷规则的集合，其规则确定了一个解决某一特定类型问题的操作序列。算法的规则必须满足以下 5 个特性。

- ⊙ 有穷性：对于任意一组合法的输入值，算法在执行有穷步骤之后一定能结束。即算法的操作步骤为有限个，且每步都能在有限时间内完成。
- ⊙ 确定性：对于每种情况下所应执行的操作，在算法中都有确切的规定，使算法的执行者或阅读者都能明确其含义及如何执行。并且，在任何条件下，算法都只有一条执行路径。
- ⊙ 可行性：算法中的所有操作都必须足够基本，都可以通过已经实现的基本操作运算有限次实现之。
- ⊙ 有输入：算法有零个或多个输入数据。输入数据是算法的加工对象，既可以由算法指定，也可以在算法执行过程中通过输入得到。
- ⊙ 有输出：算法有一个或多个输出数据。输出数据是一组与输入有确定关系的量值，是算法进行信息加工后得到的结果，这种确定关系即为算法的功能。

2. 算法设计目标

算法设计应满足以下 5 个目标。

- ⊙ 正确性：算法应确切地满足应用问题的需求，这是算法设计的基本目标。
- ⊙ 健壮性：即使输入数据不合适，算法也能做出适当处理，不会导致不可控结果。
- ⊙ 高时间效率：算法的执行时间越短，时间效率越高。
- ⊙ 高空间效率：算法执行时占用的存储空间越少，空间效率越高。
- ⊙ 可读性：算法表达思路清晰，简洁明了，易于理解。

如果一个操作有多个算法，显然应该选择执行时间短和存储空间占用少的算法。但是，执行时间短和存储空间占用少有时是矛盾的，往往不可兼得，此时算法的时间效率通常是首要考虑因素。