应用型本科信息大类专业“十二五”规划教材

数据结构

（C语言版）

邓 奕 王维虎 沈海龙 主 编

華中科技大學出版社
http://www.hustp.com

应用型本科信息大类专业“十二五”规划教材

数据结构(C 语言版)

主　编　邓　奕　王维虎　沈海龙
副主编　苏　艳　薛晓亚　马金霞
　　张采芳　胡广义　齐晶薇

华中科技大学出版社
中国·武汉

内 容 简 介

算法＋程序＝数据结构。数据结构是设计和实现编译程序、操作系统、多媒体信息处理、自动控制系统、数字图像处理及其他系统程序和大型应用程序的重要基础，是介于数学，计算机硬件、软件之间的一门核心课程，是计算机学科中一门综合性的专业基础课。

全书分为八章，主要包括绪论，线性表，栈和队列，串、数组和广义表，树，图，查找及排序等。本书介绍了各种基本类型的数据结构及其应用，重点讨论了查找的各种实现方法。全书采用C语言作为数据结构和算法的描述语言。

本书可作为高等院校计算机类、电子信息类、自动化类、电气类、光电类及其他相关专业本专科学生的教材，也可作为工程技术人员的参考资料和感兴趣的读者的自学读物。

为了方便教学，本书还配有电子课件等教学资源包，相关教师和学生可以登录“我们爱读书”网(www.ibook4us.com)免费注册下载，或者发邮件至hustpeiit@163.com免费索取。

图书在版编目(CIP)数据

数据结构：C语言版/邓奕，王维虎，沈海龙主编. —武汉：华中科技大学出版社，2014.6
应用型本科信息大类专业“十二五”规划教材
ISBN 978-7-5680-0141-0

Ⅰ.①数… Ⅱ.①邓… ②王… ③沈… Ⅲ.①数据结构-高等学校-教材 Ⅳ.①TP311.12

中国版本图书馆CIP数据核字(2014)第118710号

数据结构(C语言版) 邓 奕 王维虎 沈海龙 主编

策划编辑：康 序
责任编辑：史永霞
封面设计：李 嫚
责任校对：祝 菲
责任监印：张正林
出版发行：华中科技大学出版社(中国·武汉)
武昌喻家山 邮编：430074 电话：(027)81321913
录 排：武汉正风天下文化发展有限公司
印 刷：武汉市籍缘印刷厂
开 本：787mm×1092mm 1/16
印 张：12.75
字 数：331千字
版 次：2015年8月第1版第1次印刷
定 价：35.00元

前言 PREFACE

算法＋程序＝数据结构。数据结构是设计和实现编译程序、操作系统、多媒体信息处理、自动控制系统、数字图像处理及其他系统程序和大型应用程序的重要基础，是介于数学和计算机硬件、软件之间的一门核心课程，是计算机学科中一门综合性的专业基础课。

“数据结构”是计算机程序设计的重要理论基础，它不仅是计算机、软件工程等专业的核心课程，而且因为其特点和重要性，已成为其他理工科专业的热门选修课。全书分为八章，主要包括绪论，线性表，栈和队列，串、数组和广义表，树，图，查找及排序等。本书介绍了各种基本类型的数据结构及其应用，重点讨论了查找的各种实现方法。全书采用C语言作为数据结构和算法的描述语言。全书内容选取既符合教学大纲要求，又兼顾学科的广度和深度，适用面广。

本书由汉口学院邓奕和王维虎、大连理工大学城市学院沈海龙担任主编，由华中师范大学武汉传媒学院苏艳、青岛理工大学琴岛学院薛晓亚和马金霞、文华学院张采芳和胡广义、哈尔滨远东理工学院齐晶薇担任副主编。其中：邓奕副教授制定了本书的编写大纲，对全书进行了统稿，并具体编写了第1章；王维虎编写了第2章；沈海龙编写了第5章；苏艳编写了第3章；薛晓亚编写了第4章；马金霞编写了第6章；张采芳编写了第7章；胡广义和齐晶薇编写了第8章。

在撰写本书期间，编者得到了前辈、家人、同事、朋友的支持和帮助，在此深表感谢。

本书可作为高等院校计算机类、电子信息类、自动化类、电气类、光电类及其他相关专业学生的教材和教学参考书，也可作为工程技术人员的参考资料和感兴趣的读者的自学读物。为了方便教学，本书还配有电子课件等教学资源包，相关教师和学生可以登录“我们爱读书”网（www.ibook4us.com）免费注册下载，或者发邮件至hustpeiit@163.com免费索取。

由于时间仓促，书中难免有疏漏之处，请读者谅解。如果读者在学习、实践或者教学过程中遇到问题，可通过402345008@qq.com与我们交流。

编　者

2015年3月

目录 CONTENTS

第1章 绪 论

现今，计算机的应用已渗透到各个领域，对于计算机处理的数据也不再只有单一的形式，而是各种具有一定结构的数据。因此，“数据结构”不仅是计算机科学与工程的基础研究之一，也是其他非计算机专业的核心课程之一，它贯穿程序设计的始终，掌握该领域的知识对我们进一步学习计算机程序开发非常重要，也是设计和实现编译程序、操作系统、数据库系统和各种大型应用程序的重要基础。

1.1 数据结构简介

数据结构在计算机科学界至今没有一个标准的定义。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。” Clifford A. Shaffer 在《数据结构与算法分析》一书中给出的定义是：“数据结构是 ADT（抽象数据类型 abstract data type）的物理实现。” Robert L. Kruse 在《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。

在计算机科学或信息科学中，数据结构（data structure）是计算机中存储、组织数据的方式。我们可以简单地理解为，数据结构代表了数据元素之间的一种或多种关系集合。根据元素之间的关系，我们可以想到线性结构、树形结构和网状结构这几种基本的结构。下面我们举些例子说明这些结构。

例 1-1 学生成绩查询系统。

某学校有 N 个学生，每个学生有他们相应的学号，某系统里存储了他们某科各自的考试成绩。假定按照如下形式安排：$(a_1,b_1),(a_2,b_2),\cdots,(a_n,b_n)$，其中 $a_i,b_i(i=1,2,\cdots,n)$ 分别表示某学生的学号和考试成绩。本问题就是一种最简单的一一对应问题，如表 1-1 所示，数据与数据之间就构成了简单的一对一的线性关系。

表 1-1 线性表结构

姓 名	学 号	成 绩
张三	90123001	85
李四	90123002	93
王五	90123003	78
⋮	⋮	⋮

例 1-2 磁盘目录文件系统。

磁盘根目录下面有很多子目录及文件，每个子目录又可以包含若干个子目录及文件。本问题就是一种典型的树形结构问题，如图 1-1 所示。

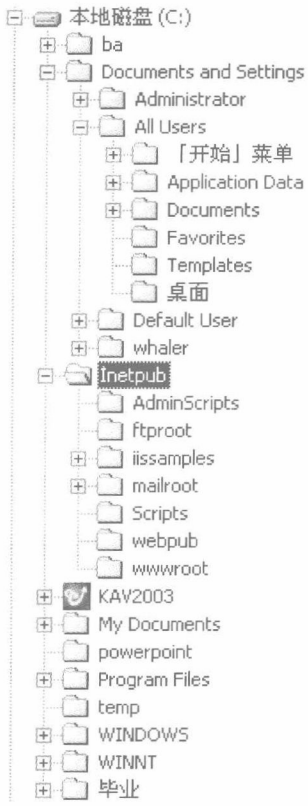


图 1-1　树形结构

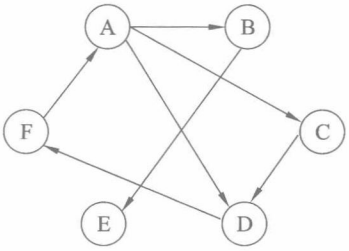


图 1-2　网状结构

例 1-3　人际关系。

在社交软件中，通常有这么一个功能：提示你可能认识的人。在社会中，人与人的关系是复杂的，如 A 和 B、C 是好朋友，C 又认识 D，那么 A 就有可能会认识 D。图 1-2 所示就是一个网状结构，其数据与数据成多对多的关系，是一种非线性关系结构。

不同种类的数据结构适用于不同种类的应用，其中部分甚至应用于某些特定的方向。在程序设计中，选择适当的数据结构是一个主要的考虑因素。系统实现的困难大小和系统构造的质量好坏都依赖于是否选择了一个最优的数据结构。一个设计良好的数据结构，应该尽可能在较少的时间和空间资源的前提下，为各种运行提供支持。

1.2　基本概念和术语

下面要介绍的一些基本概念和术语是本书中频繁出现的，我们将给以明确的定义。

1. 数据

数据(data)：是客观事物的符号表示。在计算机科学中数据指的是所有能输入到计算机中并被计算机程序处理的符号的总称。

例如，张三同学数学考试成绩为 85 分，那么 85 就是该同学的成绩数据；再例如，图像、声音等，这些都可以是数据。首先要将信息符号化，才便于我们的处理，尤其是便于计算机的处理。

2. 数据元素

数据元素(data element):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。数据元素也可以再由不可分割的数据项组成。我们回到表 1-1,整个表记录的是学生成绩数据,单个学生成绩是其中的一个数据元素。“张三、90123001、85”这三个一起构成一个数据元素,而“85”只是一个数据项。

3. 数据对象

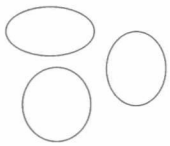
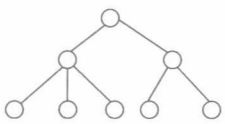
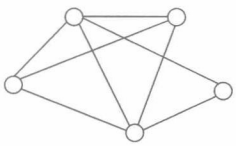
数据对象(data object):是性质相同的数据元素的集合,是数据的一个子集。

如例 1-1,一个班的成绩表可以看作一个数据对象。

4. 数据结构

数据结构(data structure):相互之间具有一定联系的数据元素(也可称数据对象)的集合。元素之间的相互联系称为逻辑结构。数据元素之间的逻辑结构有四种基本类型,如表 1-2 所示。

表 1-2 数据结构的四种基本类型

类 型	特 征	示 例
集合	元素间为 仅同属一个集合	
线性结构	元素间为严格的 一对一的关系	表 1-1
树形结构	元素间为严格的 一对多的关系	
网状结构 (或图状结构)	元素间为 多对多的关系	

数据结构的形式定义是一个二元组:

$$Data\_Structure=(D,R)$$

其中:D 是数据元素的有限集,R 是 D 上关系的有限集。

例如,用图形表示出下列数据结构。

- S=(D,R)
- D={ a,b,c,d,e,f }
- R={(a,e),(b,c),(c,a),(e,f),(f,d)}

上述表达式可用图形表示为 $b\rightarrow c\rightarrow a\rightarrow e\rightarrow f\rightarrow d$,此结构为线性的。

1.3 抽象数据类型

1. 数据类型

数据类型是指一组性质相同的值的集合及定义在此集合上的一些操作的总称。

在C语言中，每个变量、常量和表达式都有各自的取值范围，类型就是用来说明变量或表达式的取值范围和所能进行的操作的。

按照取值的不同，数据类型可以分为以下两大类。

（1）基本类型：是不可再继续分割的基本类型，包括整型、实型、字符型等。

（2）构造类型：是由若干个类型组合而成的，是可以再分割的。例如，整型数组是由若干整型数据组成的。

程序语言最终是通过编译器或解释器换成底层语言的，比如汇编语言甚至是机器语言的数据类型来实现的。当我们用高级语言编写程序需要完成一个 $a+b$ 的整型运算时，我们是不需要关心在计算机内部是如何表示的，也不用知道这些操作是如何实现的，这些对于高级语言来说根本不重要。所以，无论是什么语言，都会面临整数运算、实数运算、字符运算等操作，我们可以考虑把它们都抽象出来。

2. 抽象数据类型

抽象数据类型(abstract data type，ADT)指的是一个数学模型及定义在该模型上的一组操作数。抽象数据类型的定义仅是一组逻辑特性描述，与其在计算机内的表示和实现无关。因此，不论 ADT 的内部结构是怎么样变化的，只要其数学特性不变，都不会影响其外部使用。

事实上，ADT 和数据类型实质上是一个概念。其区别是：ADT 的范畴更广，它不再局限于系统已定义并实现的数据类型，还包括用户自己定义的数据类型。

抽象数据类型最重要的特点是体现了程序设计中的问题分解、抽象和信息隐藏的特性。抽象数据类型把实际生活中的问题分解为许多个小规模且容易处理的问题，然后建立一个计算机能处理的数据模型，并把每个功能模块的实现细节作为一个独立的单元，这样就可以把实现的过程隐藏起来。

下面给出描述抽象数据类型的标准格式：

```
ADT  <抽象数据类型名> {
数据对象：<数据对象的定义>
数据关系：<数据关系的定义>
基本操作：<基本操作的定义>
} ADT <抽象数据类型名>
```

其中数据对象和数据关系的定义用伪码描述。

基本操作的定义是：

```
<基本操作名> (<参数表>)
初始条件：<初始条件描述>
操作结果：<操作结果描述>
```

其中：初始条件是操作执行之前数据结构和参数应该满足的条件，如果不满足条件，则操作失败，返回相应的错误信息；操作结果是描述操作正常完成之后，数据结构的变化情况和应返回的结果。

1.4 C语言基础

本节对C语言做一个简单的回顾。

1. 预处理命令

凡是以“#”开头的命令均是预处理命令。“define”为宏定义命令。无参宏的宏名后不带参数，其定义的一般形式为：

```
#define  标示符  字符串
```

此外，我们常对程序中反复使用的表达式进行宏定义，例如：

```
#define Q (x*x+3)
```

宏定义允许嵌套，在宏定义的字符串中可以使用已经定义的宏名，在宏展开时由预处理程序层层代换。

我们需要注意，宏定义只是简单的字符串代换，是在预处理时完成的，而typedef是在编译时处理的，它不是简单的代换关系，而是对类型说明符重新命名，被命名的标示符具有类型定义说明的功能。例如：

```
#define PORT1 int*
typedef  (int*) PORT2;
```

从形式上看，这两者是相似的，但是实际使用上它们却有差别。

PORT1 a,b; 在宏代换后变成:int *a,b;

表示a是指向整型的指针变量，而b是整型变量。然而，PORT2 a,b;表示a,b都是指向整型的指针变量，因为PORT2是一个类型说明符。

2. 函数

从函数定义的角度看，函数可分为库函数和用户定义函数两种。库函数是由C系统提供的，无须用户定义，也不必在程序中做类型说明，只需在程序前包含有该函数原型的头文件即可在程序中直接调用。而用户定义函数需要由用户按照需要来写。对于用户定义函数，不仅需要在程序中定义函数本身，而且在主函数模块中还必须对该被调函数进行类型说明，然后才能使用。

1）无参函数定义的一般形式

```
类型标识符  函数名()
{  声明部分
语句 }
```

2）有参函数定义的一般形式

```
类型标识符 函数名(形式参数表列)
{   声明部分
语句 }
```

3）函数调用的一般形式

```
函数名(实际参数表)
```

对于无参函数调用时则无实际参数表。实际参数表中的参数可以是常数、变量或其他构造类型的数据及表达式。各实参之间用逗号分隔。

3. 选择语句

条件语句 1　if（表达式）语句；
条件语句 2　if（表达式）语句；
　　else 语句；
开关语句 1　switch（表达式）{
case 值 1：语句 1；break；
……
case 值 n：语句 n；break ；
default ：语句 n＋1；｝
开关语句 2　switch {
case 值 1：语句 1；break；
……
case 值 n：语句 n；break ；
default ：语句 n＋1；｝

4. 循环语句

for 语句　　for（赋初值表达式序列；条件；修改表达式序列）语句；
while 语句　　while（条件）语句；
do-while 语句　do{
语句序列；
}while（条件）；

5. 输入输出语句

scanf（[格式串]，变量 1，…，变量 n）；
printf（[格式串]，变量 1，…，变量 n）；

6. 逻辑运算

与运算 &&、或运算 ||、非运算 ！

7. 位运算

位与 &、位或 |、位非～、位异或 ^、左移≪、右移≫

8. 赋值运算

简单赋值＝
复合算术赋值＋＝、－＝、＊＝、/＝、％＝
复合位运算赋值 &＝、|＝、^＝、≫＝、≪＝

9. 注释

单行注释 // 文字

1.5　算法和算法分析

1.5.1　算法

什么是算法呢？如今普遍认可的对算法的定义是：算法是解决特定问题求解步骤的描

述，在计算机中表现为指令的有限序列，并且每条指令表示一个或多个操作。

算法具有以下五个特性。

(1) 有穷性：一个算法必须总是在执行有穷步之后结束，且每一步都在有穷时间内完成。

(2) 确定性：算法中每一条指令必须有确切的含义，不存在二义性，且算法只有一个入口和一个出口。

(3) 可行性：一个算法是能行的，即算法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：一个算法有零个或多个输入，这些输入取自于某个特定的对象集合。

(5) 输出：一个算法有一个或多个输出，这些输出是同输入有着某些特定关系的量。

一个算法可以用多种方法来描述，主要有：使用自然语言描述；使用形式语言描述；使用计算机程序设计语言描述。

1.5.2 算法设计的要求

算法不是唯一的，也就是说，同一个问题，可以有多种解决问题的算法。尽管算法不唯一，但是相对好的算法还是存在的，掌握好的算法，对我们解决问题有很大的帮助。那么，一个相对好的算法，应该满足以下几个要求。

(1) 正确性：算法的正确性是指算法至少应该能正确反映问题的需求，具有输入、输出及加工处理无歧义性。

(2) 可读性：算法设计的另一目的是便于阅读、理解和交流。可读性好有助于人们理解算法，晦涩难懂的算法往往隐含错误，不易被发现，并且难于调试和修改。

(3) 健壮性：算法应具有容错处理，当输入数据不合法时，算法也能做出相关处理，而不是产生异常或莫名其妙的结果。

(4) 时间效率高和存储量低：时间效率指的是算法的执行时间，对于同一个问题，如果有多个算法能够解决，执行时间短的算法效率高，执行时间长的算法效率低；存储量需求指的是算法在执行过程中需要的最大存储空间，主要指算法程序运行时所占用的内存或外部硬盘存储空间。设计算法应该尽量满足时间效率高和存储量低的需求。

1.5.3 算法效率的度量

刚才已经说明了，设计算法要提高效率，缩短执行时间，算法执行时间需通过依据该算法编制的程序在计算机上运行所消耗的时间来度量，那么，我们该如何度量一个算法的执行时间呢？这里通常有两种方法。

(1) 事后统计的方法：这种方法主要是通过设计好的测试程序和数据，利用计算机的计时器对不同算法编制的程序的运行时间进行比较，从而确定算法效率的高低。

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；二是所得时间的统计量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) 事前分析估算的方法：在计算机程序编制前，依据统计方法对算法进行估算。

我们发现，一个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：算法采用的程序设计语言；编译产生的代码质量；机器执行指令的速度；问题的规模。前三者均与具体机器相关，不考虑，那么如何理解问题的规模呢？

算法中基本操作重复执行的次数是问题规模 n 的某个函数，其时间量度记作 $T(n)=O(g(n))$，称作算法的渐近时间复杂度(asymptotic time complexity)，简称时间复杂度。其中：$O()$为渐近符号。

渐进符号(O)的定义：当且仅当存在一个正的常数 C，使得对所有的 $n\geqslant n_0$，有 $f(n)\leqslant Cg(n)$，则 $f(n)=O(g(n))$。例如：$3n+2=O(n)$，因为 $3n+2\leqslant 4n$ $(n\geqslant 2)$。

时间复杂度 $T(n)$按数量级递增顺序为表 1-3 所示。

表 1-3　时间复杂度的阶

常数阶	对数阶	线性阶	线性对数阶	平方阶	立方阶	…	k 次方阶	指数阶
$O(1)$	$O(\log_2 n)$	$O(n)$	$O(n\log_2 n)$	$O(n^2)$	$O(n^3)$		$O(n^k)$	$O(2^n)$

一般地，时间复杂度常用最深层循环内的语句中的原操作的执行频度(重复执行的次数)来表示。

我们来看一个例子：

```
(1) for(i=1,i<=n; ++i)
        for (j=1; j<=n; ++j)
          {  c[i][j]=0;
             for(k=1; k<=n; ++k)
             c[i][j]+=a[i][k]*b[k][j];  }
```

上例是一个三重循环，每个循环从 1 到 n，则总次数为 $n\times n\times n=n^3$，时间复杂度为$T(n)=O(n^3)$。

```
(2) {++x; s=0;}
(3) for(i=1;i<=n; ++i){++x; s+=x;}
(4) for(i=1;i<=n; ++i)
      for(k=1;k<=n; ++k) {++x; s+=x;}
```

在以上的三个程序段中，“x 自增 1”的操作分别重复 1 次、n 次和 n^2 次，这三种分别为常量阶、线性阶和平方阶。第三个算法的执行时间随着 n 增加将远远多于前面两个。测定运行时间的最可靠的方法就是计算对运行时间有消耗的基本操作的执行次数，运行时间与这个计数成正比。

1.5.4　算法的存储空间需求

算法的空间复杂度是通过计算算法所需的存储空间大小来度量的，算法空间复杂度的计算公式记作 $S(n)=O(f(n))$，其中，n 为问题的规模，$f(n)$为语句关于 n 所占存储空间的函数。

一般情况下，一个程序在机器上执行时，除了需要存储程序本身的指令、常数、变量和输入数据外，还需要存储对数据操作的存储单元。若输入数据所占空间只取决于问题本身，和算法无关，这样只需要分析该算法在实现时所需的辅助单元即可。若算法执行时所需的辅助空间相对于输入数据量而言是个常数，则称此算法为原地工作，空间复杂度为 $O(1)$，若为一位数组 $s[n]$，则空间复杂度为 $O(n)$。

通常，我们都是用“时间复杂度”来指运行时间的需求，使用“空间复杂度”指空间需求。当不用限定词而仅使用“复杂度”时，通常都是指时间复杂度。

本章习题

1.1 什么是数据结构，数据的逻辑结构可以分为哪几种？

1.2 试述抽象数据类型的概念与程序设计语言中数据类型概念的区别。

1.3 简述算法的五个特性及对算法设计的要求。

1.4 算法分析的目的是什么，算法分析的两个主要方面又是什么？

1.5 常见的算法时间复杂度常数阶、对数阶、线性阶、平方阶和指数阶分别用大 O 记号怎么表示？

1.6 对下列用抽象数据类型表示的数据结构，试分别画出对应的逻辑结构图。

(1) A=(D,R)，其中 D={a1,a2,a3,a4}，R={ }；

(2) B=(D,R)，其中 D={a,b,c,d,e,f}，R={(a,b),(b,c),(c,d),(d,e),(e,f)}。

1.7 分析以下各程序段，并用大 O 记号表示其执行的时间复杂度。

(1)

```
i=1;k=0
while (i<n-1)
{ k=k+10*i;
  i++;
}
```

(2)

```
y=0;
while((y+1)*(y+1)<=n)
        y=y+1;
```

(3)

```
k=0;
for(i=1; i<=n; i++) {
        for(j=i; j<=n; j++)
            k++;
}
```

1.8 已知有实现同一功能的两个算法，其时间复杂度分别为 $O(2^n)$ 和 $O(n^{10})$，假设现实计算机可连续运算的时间为 10^7 秒(100 多天)，又每秒可执行基本操作(根据这些操作来估算算法时间复杂度)10^5 次。试问在此条件下，这两个算法可解问题的规模(即 n 值的范围)各为多少？哪个算法更适宜？请说明理由。

1.9 试写一算法，自大至小依次输出顺序读入的三个整数 X、Y 和 Z 的值。

习题答案

1.1 数据结构在计算机中的表示称为物理结构，又称存储结构，是逻辑结构在存储器中的映像，包括数据元素的表示和关系的表示。逻辑结构与计算机无关。

数据的逻辑结构分为线性结构和非线性结构，也可以分为集合、线性结构、树形结构和图形即网状结构。

1.2 抽象数据类型是指一个数据结构以及定义在该结构上的一组操作。程序设计语言中