第3章 数据结构

本章学习目标

- 理解数据结构的概念
- 掌握数据结构的描述方法

数据结构由数据集合、约束条件和操作组成。约束条件一般用数据之间的二元关系表示，基本操作包括向数据集合增加数据、从数据集合删除数据、判断数据集合是否包含某个数据、判断数据集合是否为空集和计算数据集合的数据个数等操作。

数据结构的实现是指如何在计算机存储器内表示数据和数据之间的关系，以及在此基础上如何高效地实现各种操作。

3.1 数据结构的基本概念

电子数字计算机自20世纪40年代问世以来，早期用于数值计算，例如，弹道计算，其特点是数据量较少，但计算复杂。随后向银行业等商业领域扩展，其特点是计算比较简单，但数据量大，而且数据之间密切关联。例如，一张转账单关联两个账号，每个账号关联一个客户。应用需求催生了对各种数据结构的研究，20世纪60年代末，计算机科学与技术专业就设置了“数据结构”课程。

数据结构（Data Structure）没有严格统一的定义。在日常生活中，经常会用到房屋结构、产业结构和分子结构等词语，这些词语都涉及结构一词。牛津字典对 structure 的解释如下：

the arrangement of and relations between the parts or elements of something complex

- the organization of a society or other group and the relations between its members, determine its working
- a building or other object constructed from several parts

结构是复杂物体组成部分的组织方式或相互之间的关系，结构将这些组成部分连接起来形成了一个有机的整体。

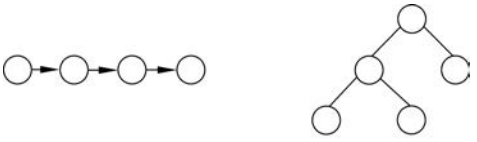
鸟巢体育馆是一个壮观的钢结构建筑，大量的钢板交织在一起，既提供了必要的支撑力，给人以美感，又使鸟巢体育馆成为著名的体育设施，如图3.1所示。

一般认为，数据结构是由数据和数据之间的关系构成的整体。

解决不同的问题需要不同的数据结构，因此，存在各种各样的数据结构。本书主要介绍

图 3.1　鸟巢体育馆

经典的数据结构，包括线性表、栈和队列等线性结构，二叉树、树等层次结构以及其他的数据结构，如图 3.2 所示。图中使用圆表示数据，使用带箭头的线段表示线性表的数据的先后关系，使用线段表示二叉树的数据的双亲/孩子关系。这些数据结构经过了深入的研究，得到了广泛的应用，部分数据结构已纳入了 Java 类库。



(a) 线性表　　(b) 二叉树

图 3.2　经典的数据结构

3.2　数据结构的描述

数据结构的描述是指如何在计算机存储器中存放数据和数据之间的关系。

1. 内存储器模型和分配方式

计算机使用存储器存储数据和程序。存储器分为内存储器和外存储器，简称内存和外存。内存由存储单元组成，存储单元有唯一的编号，叫作地址(Address)，地址从 0 开始编排，如图 3.3 所示。

内存可以执行 read、write 和 move 指令，这些指令需要明确存储单元的地址。read 指令读取存储单元存储的数据，write 指令将数据存放于存储单元(原有的数据就消失了)，move 指令将一个存储单元存储的数据复制到另一个存储单元。

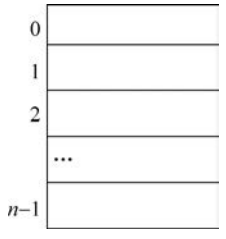


图 3.3　内存储器模型

使用机器语言，程序员必须决定将数据存储于哪个存储单元，而且必须记住存储单元的地址，以便后续对其进行读写操作。

存储大量的数据有两种内存分配(Storage Allocation)方式：**连续(顺序)分配**(Sequential Allocation)和**链式分配**(Linked Allocation)。假设内存按字节编址，long 类型的整数占用 8 字节。存储 3 个 long 型的整数 33、44、55 的两种分配方式如图 3.4 所示。

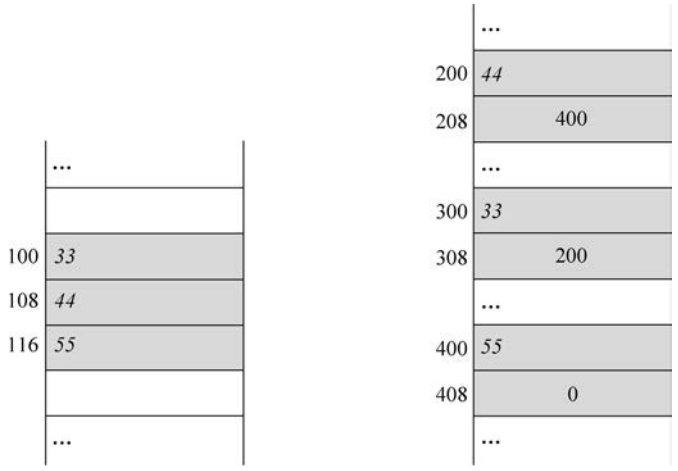


图 3.4　内存的分配方式

连续分配方式将数据分配(存储)于连续的存储单元。已知第一个数据的地址 a,就可以按照以下公式得到各数据的地址:

$$\text{location}(i)=a+(i-1)\times s,\quad 1\leqslant i\leqslant k,k \text{ 是数据个数} \tag{3.1}$$

从图 3.4(a)可知,$a=100$,$s=8$,44 是第 2 个数据,根据式(3.1),44 存储于 108 号存储单元。

连续分配方式的特点是**随机存取**(Random Access),即存取任意的数据花费相同的时间。因为按公式 3.1 就能获知其地址,然后就可对其进行读写操作。

链式分配方式将数据分配于离散的存储单元,通过附加的地址信息将数据**链接**(Link)在一起。根据第一个数据的地址 a,找到数据后,取出附加的地址信息,就能找到第二个数据,以此类推。因此,链式分配方式的特点是**顺序存取**(Sequential Access),即要读写第 i 个数据,必须先依次找到前 $i-1$ 个数据,存取不同的数据花费不同的时间。

从图 3.4(b)可知,第一个数据 33 存储于编号为 300 的存储单元,附加的地址是 200,在 200 号存储单元找到第二个数据 44,附加的地址是 400,在 400 号存储单元找到第三个数据 55,附加的地址是 0,此时可知,其后再无其他数据。因为 0 号存储单元一般受操作系统的保护,普通用户程序不能访问,所以常使用地址 0 作为最后一个数据的标志。

高级程序设计语言将存储单元抽象为变量,变量用于存储数据。高级程序设计语言通过**符号表**(Symbol Table)将变量映射到存储单元的地址,从而将对变量的读写操作转换为对存储单元的读写操作。

高级程序设计语言提供了数组用于存储大量的数据。数组由一组无名变量组成,数组将下标映射到无名变量,通过数组名和下标存取数据。数组一般用连续分配方式实现,但也可用其他方式实现。

高级程序设计语言负责内存的管理。主要的管理工作有:哪些存储单元已经被使用、哪些存储单元未被使用、为变量分配存储单元以及回收变量占用的存储单元。

有了高级程序设计语言的支持,实现数据结构就是使用变量和数组表示数据和数据之间的关系。

一般通过文件系统的文件使用外存。文件由若干字节组成，每字节有唯一的编号，这个编号是相对文件头的位置，因此，文件也构成了如图 3.3 所示的线性空间。操作系统提供了从指定位置读写文件的若干字节的操作。

在外存上实现数据结构，程序员首先要编写代码对文件的存储空间进行管理，包括分配存储空间、回收存储空间、管理已使用的空间和管理未使用的空间等功能。

2. 数据的描述

高级程序设计语言提供了丰富的**数据类型**(Data Type)用于表示各种用途的数据。数据类型由数据集合和集合上的封闭运算组成。例如常见的数据类型 int，它定义了一组整数以及整数的加、减、乘、除等运算。高级程序设计语言提供的数据类型又叫作基本数据类型。

数据有原子数据和复合数据之分。原子数据使用基本数据类型描述，复合数据由原子数据和其他复合数据组成。复合数据由用户使用高级程序设计语言提供的机制(如 C 语言的 struct、C++和 Java 语言的类)定义。

例 3.1　地址是复合数据，由街道和城市组成。

```
public class Address {
    String road;
    String city;
}
```

例 3.2　学生是复合数据，由姓名、年龄、身高和地址组成。

```
public class Student {
    String name;
    int age;
    float height;
    Address address;
}
```

类是程序员定义的数据类型。类的对象的集合等同于数据类型的数据集合，类的方法可视为数据类型的运算。

对 Java 语言而言，原子数据用基本数据类型的常量或变量表示，复合数据用对象表示。

为了通用性，本书使用了泛型，数据就是对象，对象就是数据。

3. 结构的描述

例 3.3　表 3.1 是一份学生名单。现需要将其存储到计算机，除了要保存每个学生的信息外，还要反映学生在名单中的先后关系。

表 3.1　学生名单

姓名	年龄/岁	身高/cm
甲	17	181
乙	18	170
丙	17	165

学生甲、乙、丙分列名单的第 1、2 和 3 行，他们之间有先后关系，如图 3.5 所示。

为了表示学生信息，声明 Student 类如下：

```
public class Student {
    String name;
```

图 3.5　学生在学生名单中的先后关系

```
    int age;
    float height;
}
```

生成 3 个 Student 对象用于表示 3 个学生，对象有唯一的标识，假设为 ID23、ID18 和 ID25，如图 3.6 所示。

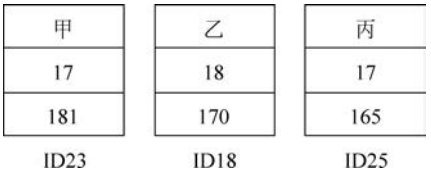


图 3.6　3 个学生对应的对象

有两种方式可以表达 3 个学生在学生名单中的先后关系。一种方式是修改 Student 类的声明，增加引用类型的字段 next，修改后的 Student 类的声明如下：

```
public class Student {
    String name;
    int age;
    float height;
    Student next;
}
```

字段 next 用于引用 Student 对象，表示先后关系。如图 3.7(a)所示，学生甲的字段 next 存储了学生乙的对象标识 ID18，表示学生甲排在学生乙的前面。学生乙的字段 next 存储了学生丙的对象标识 ID25，表示学生乙排在学生丙的前面。学生丙的字段 next 为 null，表示学生丙后面没有其他的学生，他是最后一名学生。

通过字段 next 能找到一个对象，为了更形象地表示这种含义，引入**箭头符号**，如图 3.7(b)所示。学生甲的字段 next 指向了学生乙，表示学生甲引用了学生乙，在名单中排在学生乙的前面。

这种通过引用变量表达数据之间关系的方式类似于内存的链式分配方式，故称作**链式描述**。

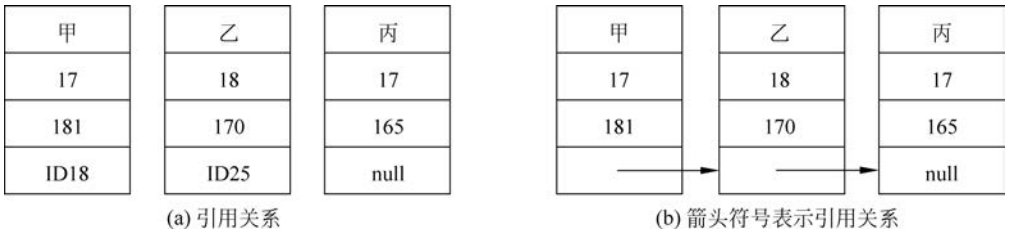


图 3.7　3 个学生对象及先后关系

为 Student 类增加字段 next 的方法很好地表达了先后关系，但有若干弊端。首先要修改 Student 类的声明，但很多情况下不能修改类声明。其次，如果学生甲出现在两个名单中，而学生乙和学生丙只出现在一个名单中，那么，为了表示学生甲在两个名单中的先后关

系，就要为 Student 类增加两个字段，但学生乙和学生丙只使用了其中的一个，浪费存储空间。

更好的方式是像 Java 类库那样引入泛型类 Node，其声明如下：

```
public class Node<T>{
    T data;
    Node<T> next;
}
```

由于字段 next 的存在，Node 对象之间存在引用关系。借助 Node 对象之间的引用关系间接地表达学生对象之间的先后关系，如图 3.8 所示。

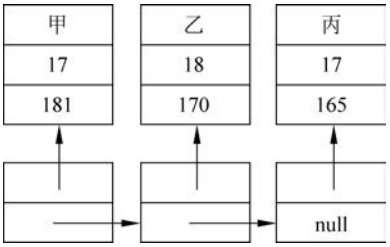


图 3.8 借助 Node 对象表示 3 个学生对象的先后关系

请注意，本书针对不同的数据结构声明了不同的 Node 类用于存储数据和数据之间的关系，为了叙述方便，将各类 Node 对象统称为**结点**。

另一种方式是使用数组，通过数组元素下标之间的次序关系间接地表示学生对象之间的先后关系，如图 3.9 所示。下标为 0、1、2 的数组元素分别引用了学生甲、乙和丙，根据下标的次序关系，学生甲先于学生乙，学生乙先于学生丙。

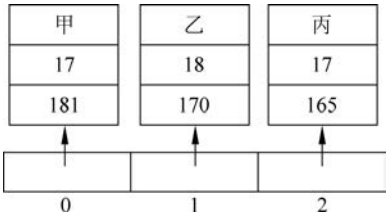


图 3.9 数组元素下标的次序关系表示学生对象的先后关系

本书将使用数组下标之间的次序关系描述数据之间关系的方式叫作**数组描述**。

3.3 抽象数据类型及实现

抽象数据类型(Abstract Data Type)由数学模型以及模型上的操作组成。例如，离散数学的图以及求顶点的出度、入度等操作就构成了一个抽象数据类型。

抽象数据类型和高级程序设计语言的基本数据类型是同一个概念，只是前者是从实际问题抽象出来的数据类型，它是程序员定义的数据类型，后者是高级程序设计语言所提供的数据类型。

高级程序设计语言的数据类型由基本数据类型和抽象数据类型组成。

抽象数据类型等同于 3.1 节介绍的数据结构及相关的操作，因此本节也可叫作数据结

构的实现。

本书使用接口定义抽象数据类型，接口规定了抽象数据类型的约束条件和操作。例如，以下的 IList 接口定义了线性表，它声明了若干抽象方法，方法的参数出现了数据编号 index，数据编号体现了数据之间的线性次序关系。

```
public interface IList<T> {
    void clear();
    boolean isEmpty();
    int size();
    T get(int index);
    void add(int index, T x);
    T remove(int index);
    T set(int index, T x);
    int indexOf(T x);
    Iterator<T> iterator();
}
```

抽象数据类型的实现就是综合运用数组描述和链式描述表达数据和数据之间的关系，在此基础上高效地实现各种操作。使用类实现抽象数据类型，例如，CLinkedList 类实现了 IList 接口，其声明如下：

```
public class CLinkedList<T> implements IList<T> {
…
}
```

由于 Java 类库已有 List 接口和 LinkedList 类，为了避免重名引起的混乱，本书在接口和类名前分别使用字母 I 和 C 作为前缀，以示区别。

小结

数据结构的核心是结构，通过结构将数据融合在一起，构成一个有机的整体。例如，线性表的数据之间的线性次序关系将数据联系起来使数据排列成直线。

数据结构有两种基本的描述方式。一种是显式的方式，即使用变量（如 C 语言的指针、Java 语言的引用变量）描述数据之间的关系。另一种是隐式的方式，即通过数组下标的次序关系描述数据之间的关系。总之，就是使用一组变量和数组描述数据结构。

需要指出的是，不同的教材使用了不同的术语。参考文献[1]的信息结构部分成文于 1968 年，没有区分数据结构的抽象和数据结构的描述，术语结点（Node）既指数据，也指数据在内存的表示。

参考文献[3]、[4]、[6]将数据结构分为逻辑结构和存储（物理）结构，例如，线性表是逻辑结构，线性表在内存的链式描述链表是存储结构。顺序结构和链式结构是两种基本的存储结构，存储结构即本书的数据结构的描述。

参考文献[7]使用了数据类型、抽象数据类型和数据结构 3 个概念，抽象数据类型和数据结构基本等同于逻辑结构和存储结构。将数据结构分为抽象与实现体现了软件分层的思想，有利于软件的开发和维护。本书将数据结构抽象的部分叫作抽象数据类型，实现的部分

叫作抽象数据类型的实现或数据结构的实现，二者分别对应 Java 的接口和类。

参考文献[3]、[6]、[8]、[10]中将数据集合叫作数据对象，集合的元素叫作数据元素，构成数据元素的字段叫作数据项。由于本书使用 Java 描述数据结构，故只使用了数据的概念，数据就是对象，对象就是数据。

习题

1. 填空题

(1) 连续分配方式的特点是_____，存取不同的数据花费_____的时间。

(2) 链式分配方式的特点是_____，存取不同的数据花费_____的时间。

(3) 原子数据使用_____表示，复合数据使用_____表示。

(4) 表达数据之间的关系的方式有_____和_____，其中，最基本的方式是_____。

(5) 数组由一组_____构成，通过_____存取数组存储的数据。

(6) 本书使用_____表示抽象数据类型，使用_____表示数据结构。

2. 论述题

(1) 使用高级程序设计语言表示数据之间的关系的方法有几种？各自的特点是什么？

(2) 泛型类 Node 的作用是什么？

(3) 如果要在文件中实现数据结构，程序员需要首先设计和实现什么？

(4) 将数据结构划分为抽象和实现有什么优点？