第3章 数据表示

在计算机发展的早期，编写程序的目的是为了完成对数值数据的计算。随着计算机技术的不断发展和应用范围的拓展，数据处理成为计算机的一个重要应用领域，此时的数据包括数值数据，也包括非数值数据（如字符串、图像等），处理既可以是算术运算，也可以是插入、删除、查找和排序等操作。

在计算机系统上处理数据，需要解决三个问题：用什么方法表示数据？用什么方法表示数据的加工过程？前两个表示怎么在机器上实现？本章主要解决第一个问题，在计算机内用分层次的方法来表示数据。

3.1 数据的分层表示

计算机科学用数据来表示客观世界里要处理的对象。计算机只能用二进制来存储、处理和传送各类信息，即能够物理实现的数据记号是二进制数字“0”和“1”，因此数据表示面临的任务是用最简单的记号表示内容复杂而形式多变的对象。可以把计算机科学在不同时期提出的数据表示方法，总结为一种分层次的表示数据的方法，图 3.1 描述了这种层次。

3.1.1 现实世界层

现实世界层的数据就是从现实世界客观事物中提取出来的一组特征，也就是说，不管事物是有形的还是无形的，总是用事物特征的一个集合来表示事物本身。提取事物特征的原则有：抓主要矛盾、抓重点、真实、准确、够用。如下所示学生的信息，在不同应用中的特征有所不同。

教学系统：学号、姓名、班级、专业、政治、英语等。

教务系统：学号、姓名、性别、年龄、身高、政治面貌等。

3.1.2 信息世界层

信息世界层中一种事物的特征几乎有无限多个，在处理事物时需要选择恰当的特征来代表事物；而且，在处理数据任务时有可能要面对多种事物，此时需要抽象出数据对象之间的关联，以便组成一个统一的数据表示结构，称之为信息结构。在众多信息结构中，实体-联系模型和数据结构是两种被广泛应用的结构。

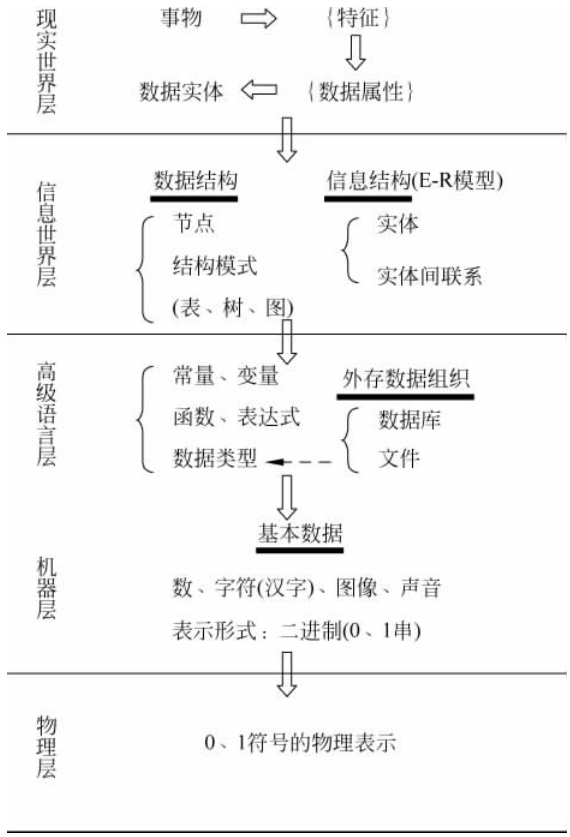

图 3.1 数据的分层表示

1. 实体-联系(ER)模型

实体-联系模型有三个组成部分，即实体(Entity，实体集)、联系(Relationship，联系集)和属性(Attribute)。实体描述的是现实世界中的对象或概念，实体集是具有相同类型和性质(属性)的实体集合；联系刻画实体之间的关联情况，联系集是同类联系的集合；属性是实体的性质和特征。

ER 模型一般用 ER 图来表示，图中实体用方框表示，联系用菱形框表示，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型，属性用椭圆框表示；在框中标注实体名、联系名、属性名，如图 3.2 所示。

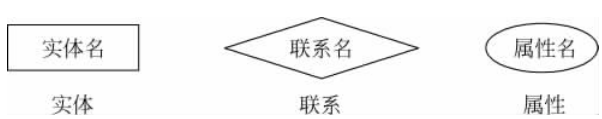

图 3.2 ER 图常用符号

实体之间的联系有多种类型，一般两个实体之间联系即二元联系有三种类型(1∶1，1∶n 或 m∶n)，如图 3.3 所示。

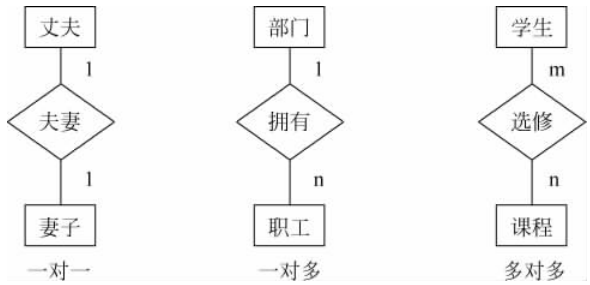

图 3.3　二元联系示意图

【例 3.1】 用 ER 图表示课程和教室的 ER 模型。

解：若不需要考虑教室的特性，则教室可作为课程的属性；否则就应把教室作为实体。该模型的 ER 图如图 3.4 所示。

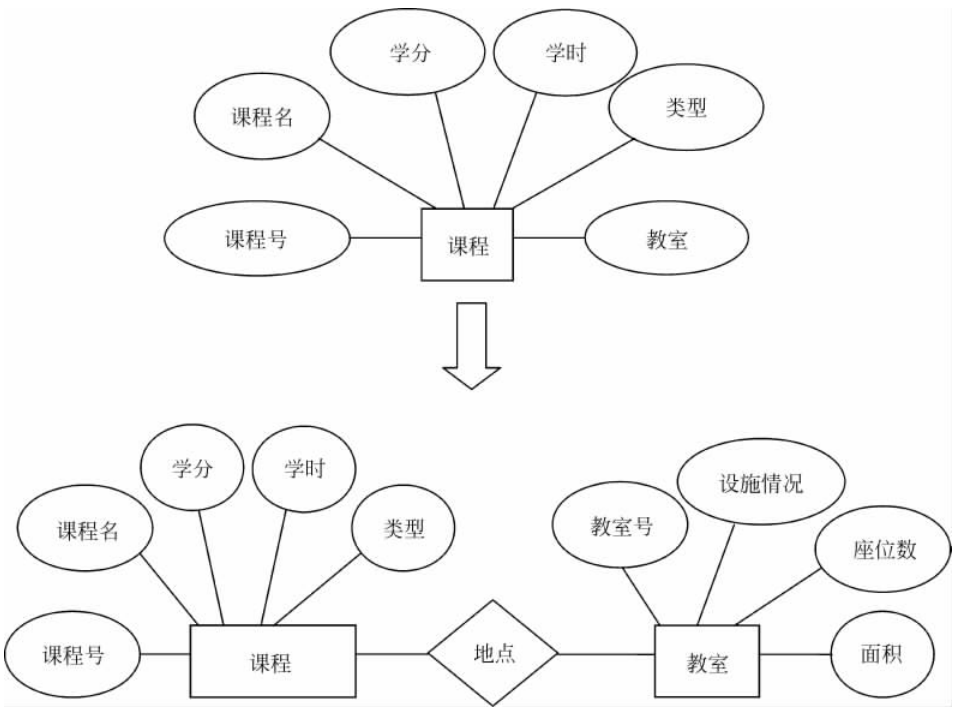

图 3.4　课程和教室的 ER 图

2. 数据结构

数据结构(Data Structure)简称 DS，是数据元素的组织形式，或数据元素之间存在的一种或多种特定关系的集合。任何数据都不是彼此孤立的，通常把相关联的数据按照一定的逻辑关系组织起来，按照计算机语言中语法、语义的规定将结构或形式进行相应存储，并且为这些数据指定一组操作，这样就形成了一个数据结构。

程序要处理的数据需要存储在内存单元中，整个内存空间是由连续编址的一个个内存单元组成的，就是说，多么复杂的数据也只能存放在这样的一个空间内，这是数据存储的物理结构。对于一些简单的运算，编程人员可直接面对这种存储方式，例如，在机器语言和汇

编语言编程阶段，程序员就直接使用这种物理存储结构。这种方式的数据处理能力有限、编程复杂，对于矩阵、家族关系表这样的数据就会增加编程人员的编程难度和工作量。能否找到一种机制，使得数据的内部存储结构（物理结构）是线性连续的，而其逻辑结构更符合人们习惯的方式，编写程序时面对的是逻辑结构（数据的一种抽象方式），程序执行时由支持相应结构的编译程序自动把逻辑结构映射成物理结构，从而简化程序的编写，减轻编程人员的工作量，这就是数据结构要解决的问题之一。

3.1.3 高级语言层

信息世界层的数据是抽象的，表示数据的概念和方法仍然没有进入到计算机系统的范围。而高级语言层就是进一步将信息世界层抽象的信息结构利用程序设计语言提供的数据表达手段来表示，如此更加接近数据表示的最终目标。

目前为止，高级语言是程序设计的主要工具。尽管有多种高级语言，但是数据对象的表示可以归纳成常量、变量、表达式、函数和数据类型五种方式。其中，常量和变量是数据表示的基本概念，表达式和函数表示经过操作过程得到的结果数据，数据类型刻画不同种类的数据。

3.1.4 机器层

高级语言层的数据表示方式已经进入计算机系统范围，经过编译，高级语言层的数据可以转换成机器层的数据。机器层靠各种数据编码规则将二进制数存储在顺序组织的、可寻址的存储单元中。

机器层的数据按照基本用途可以分为数值型数据和非数值数据两类。数值型数据表示具体的数量，有正负大小之分。非数值型数据主要包括字符、声音、图像等，这类数据在计算机中存储和处理前需要以特定的编码方式转换为二进制形式。

3.1.5 物理层

物理硬件是数据表示的最终层次，数据表示的目标就是物理元件以两个稳定的并可以按照需求相互转换的物理状态表示二进制数字 0 和 1。比如，开关的接通和断开、晶体管的导通和截止、磁元件的正负剩磁、电位电平的高与低等都可表示 0 和 1 两个数码。因此，电子器件具有实现二进制的可行性。

3.2 信息世界层的数据表示

实体-联系模型和数据结构是信息世界层中两种被广泛应用的信息结构。本部分重点介绍数据结构。

3.2.1 数据结构定义

用计算机求解问题首先要抽象出问题的模型，对于数值问题抽象出的模型通常是数学方程，对于非数值问题抽象出的模型通常是表、树或图等数据结构。

数据结构是计算机存储和组织数据的方式。数据结构是指相互之间存在一种或多种特定关系的数据元素的集合。通常情况下，精心选择的数据结构可以带来更高的运行或者存储效率。数据结构往往同高效的检索算法和索引技术有关。

数据结构由相互之间存在着一种或多种关系的数据元素的集合和该集合中数据元素之间的关系组成。记为：

```
Data_Structure = (D,R)
```

其中，D 是数据元素的集合，R 是该集合中所有元素之间的关系的有限集合。

数据结构是数据的组织、存储和运算的总和。它是信息的一种组织方式，是按某种关系组织起来的一批数据，其目的是为了提高算法的效率，然后用一定的存储方式存储到计算机中，并且通常与一组算法的集合相对应，通过这组算法集合可以对数据结构中的数据进行某种操作。计算机处理的大量数据都是相互关联，彼此联系的。

3.2.2　数据抽象

1. 数据

数据即信息的载体，是对客观事物的符号表示，指能输入到计算机中并被计算机程序处理的符号的总称，如整数、实数、字符、文字、声音、图形和图像等都是数据。

2. 数据元素

数据元素是数据的基本单位，它在计算机处理和程序设计中通常作为一个整体进行考虑和处理。数据元素一般由一个或多个数据项组成，一个数据元素包含多个数据项时，常称为记录和节点等，数据项也称为域、段、属性、表目和顶点等。

3. 数据对象

数据对象是具有相同特征的数据元素的集合，是数据的一个子集。

4. 数据的逻辑结构

数据的逻辑结构指数据结构中数据元素之间的逻辑关系，它是从具体问题中抽象出来的数学模型，是独立于计算机存储器的（与具体的计算机无关）。

数据的逻辑结构通常有集合形式、线性结构、树型结构、图形结构或网状结构四类基本形式。

5. 数据的存储结构

数据的存储结构是数据的逻辑结构在计算机内存中的存储方式，又称物理结构。数据存储结构要用计算机语言来实现，因而依赖于具体的计算机语言。数据存储结构的特点是用数据元素在存储器的相对位置来体现数据元素相互间的逻辑关系，有顺序和链式两种不同的方式。

顺序存储方式是把逻辑上相邻的节点存储在物理位置相邻的存储单元里，节点间的逻辑关系由存储单元的邻接关系来体现，由此得到的存储表示称为顺序存储结构。顺序存储

结构是一种最基本的存储表示方法，通常借助于程序设计语言中的数组来实现。

链接存储方式不要求逻辑上相邻的节点在物理位置上亦相邻，节点间的逻辑关系是由附加的指针字段表示的，由此得到的存储表示称为链式存储结构。链式存储结构通常借助于程序设计语言中的指针类型来实现。

在顺序存储结构的基础上，又可延伸变化出索引存储和散列存储。

索引存储就是在数据文件的基础上增加了一个索引表文件，通过索引表建立索引，可以把一个顺序表分成几个顺序子表，其目的是提高查找效率，避免盲目查找。

散列存储就是通过数据元素与存储地址之间建立起的某种映射关系，使每个数据元素与每一个存储地址之间尽量达到一一对应的目的。这样，查找时同样可以大大地提高效率。

6. 数据类型

数据类型是一组具有相同性质的操作对象和该组操作对象上的运算方法的集合，如整数类型、字符类型等。每一种数据类型都有具备自身特点的一组操作方法(即运算规则)。

7. 抽象数据类型

抽象数据类型是指一个数据模型以及在该模型上定义的一套运算规则的集合。在对抽象数据类型进行描述时，要考虑到完整性和广泛性，完整性就是要能体现所描述的抽象数据类型的全部特性，广泛性就是所定义的抽象数据类型适用的对象要广。它与数据类型实质上是一个概念，但其特征是使用与实现分离，实行封装和信息隐蔽(独立于计算机)。

3.2.3 线性结构

线性结构是数据元素之间定义了次序关系的集合(全序集合)，描述的是一对一关系。线性结构要满足几个条件：①有且只有一个根节点；②每一个节点最多有一个前驱节点，也最多有一个后继节点；③首节点无前驱节点，尾节点无后继节点。注意：在一个线性结构中插入或删除任何一个节点后还应是线性结构；否则，不能称为线性结构。常见的线性结构有数组和线性表等。

1. 数组

在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来，这些按序排列的同类数据元素的集合称为数组。在 C 语言中，数组属于构造数据类型，一个数组可以分解为多个数组元素，这些数组元素可以是基本数据类型或是构造类型。因此按数组元素的类型不同，数组又可分为数值数组、字符数组、指针数组和结构数组等各种类别。

数组是 n 个类型相同的数据元素构成的序列，它们连续存储在计算机的存储器中，且数组中的每个元素占据相同的存储空间。

【例 3.2】 下面是定义的几种数组例子。

int a[10]：定义整型数组 a，有 10 个元素。

float b[10],c[20]：定义实型数组 b，有 10 个元素；实型数组 c，有 20 个元素。

char ch[20]：定义字符数组 ch，有 20 个元素。

int a[3][4]：定义了一个三行四列的整型数组，数组名为 a。该数组共有 12 个元素，即

a[0][0],a[0][1],a[0][2],a[0][3]

a[1][0],a[1][1],a[1][2],a[1][3]

a[2][0],a[2][1],a[2][2],a[2][3]

对数组的描述通常包含下列 5 种属性。

- 数组名称：声明数组第一个元素在内存中的起始地址；
- 维度：每一个元素所含数据项的个数，如一维数组、二维数组等；
- 数组下标：元素在数组中的存储位置；
- 数组元素个数；
- 数组类型：声明此数组的类型，它决定数组元素在内存所占有的空间大小。

大多数情况下，数组下标是介于 0～$n-1$ 或 1～n 的整数，只要指定数组的下标就能够访问这些元素。对数组的常见操作包括插入、删除、排序和查找等。

2. 线性表

线性表由一组数据元素构成，数据元素的位置取决于自己的序号，元素之间的相对位置是线性的。非空线性表的结构特征：①有且只有一个根节点 a_1，它无前驱节点；有且只有一个终端节点 a_n，它无后继节点；②除根节点与终端节点外，其他所有节点有且只有一个前驱节点，也有且只有一个后继节点。节点个数 n 称为线性表的长度，当 $n=0$ 时，称为空表。在线性表上常用的运算包括初始化、求长度、取元素、修改、插入、删除、检索和排序。图 3.5 是线性表示意图，其中 t_{n-1} 表示第 n 个元素与第 1 个元素位置的距离。

栈和队列是两种运算时受到某些特殊限制的线性表，故也称为限定性的数据结构。

(1) 栈

栈是只能在某一端插入和删除的特殊线性表。它按照先进后出的原则存储数据，先进入的数据被压入栈底，最后的数据在栈顶，需要读数据的时候从栈顶开始弹出数据(最后一个进栈数据被第一个读出来)，图 3.6 是栈的示意图。栈对于实现递归算法是不可缺少的数据结构。

(2) 队列

队列是一种特殊的线性表，按照“先进先出”或“后进后出”的原则组织数据，它只允许在表的前端(Front)进行删除操作，而在表的后端(Rear)进行插入操作。进行插入操作的端称为队尾，进行删除操作的端称为队头，图 3.7 是队列的示意图。队列中没有元素时，称为空队列。

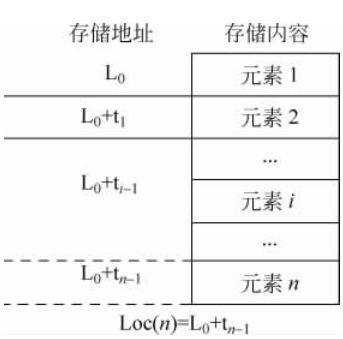

图 3.5　线性表示意图

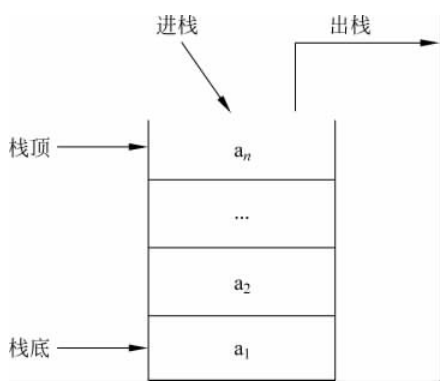

图 3.6　栈的示意图

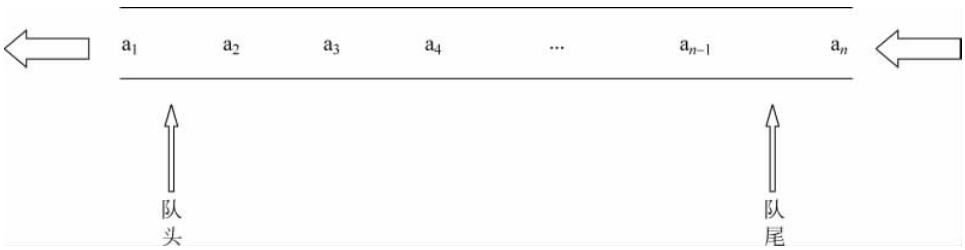

图 3.7 队列示意图

3.2.4 树形结构

树形结构是数据元素之间定义了层次关系的集合(偏序集合),描述的是一对多关系。树是包含 $n(n>0)$ 个节点的有穷集合 K,且在 K 中定义了一个关系 N,N 满足以下条件:

(1) 有且仅有一个节点 K_0,它对于关系 N 来说没有前驱节点,称 K_0 为树的根节点,简称为根(root)。

(2) 除 K_0 外,K 中的每个节点,对于关系 N 来说有且仅有一个前驱节点。

(3) K 中各节点,对关系 N 来说可以有 m 个后继节点($m\geqslant 0$)。

图 3.8 是一个树形结构的例子,有 12 个节点,A 是根节点,该树又可再分为若干不相交的子树,如 T1={B,E,F,K},T2={C,G},T3={D,H,I,J,L}等。

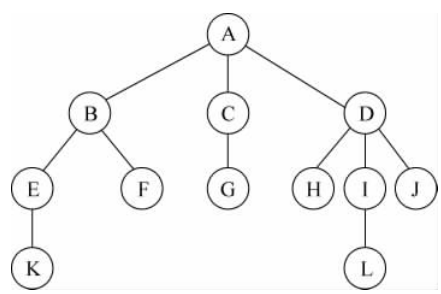

图 3.8 树形结构

3.2.5 图形结构

图形结构是数据元素之间定义了网状关系的集合,描述的是多对多关系。图由节点的有穷集合 V 和边的集合 E 组成。为了与树形结构加以区别,在图结构中常常将节点称为顶点,若两个顶点之间存在一条边,就表示这两个顶点具有相邻关系。

图的形式化定义为:

$$G=<V,E>$$

其中,V 是一个非空节点的集合;E 是连接节点的边的集合。

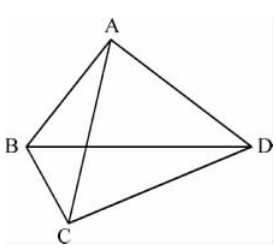

图 3.9 图形结构

【例 3.3】 图 3.9 是一个图例子,G=<V,E>。

其中,

$$V=\{A,B,C,D\},$$

$$E=\{(A,B),(A,C),(B,D),(B,C),(D,C),(A,D)\}$$

计算机算法中,图主要有邻接矩阵和邻接链表两种表示方法。

3.3 高级语言层的数据表示

3.3.1 常量

常量是在程序运行中，不会被修改的量。另一层含义指它们的编码方法是不变的，比如字符“A”无论在硬件、软件还是各种编程语言中，它的信息编码都为 0x41。

常量有不同的类型，如 25、0、－8 为整型常量，6.8、－7.89 为实型常量，“a”、“b”为字符常量。常量一般从其字面形式即可判断，这种常量称为字面常量或直接常量。

3.3.2 变量

变量一词来源于数学，在计算机语言中是储存计算结果或表示值的抽象概念，变量可以通过变量名访问。

由于变量能够把程序中准备使用的每一数据都赋予一个简短、易于记忆的名字，因此十分有用。变量可以保存程序运行时用户输入的数据、特定运算的结果以及要在窗体上显示的一段数据等。简而言之，变量是用于跟踪几乎所有类型数据的简单工具。

在高级语言中，变量是基本的数据表示方式。使用变量，程序可以表示一类数据，而不仅仅是一个数据。例如，用程序计算圆面积时，如果圆半径用常量 4.5 表示，那么只能计算一个特定的圆的面积；如果圆半径用变量 r 来表示，只需要设定 r 值，就能计算不同圆的面积。

3.3.3 函数

函数(Function)名称出自数学家李善兰的著作《代数学》。之所以如此翻译，他给出的原因是“凡此变数中函彼变数者，则此为彼之函数”，即函数指一个量随着另一个量的变化而变化，或者说一个量中包含另一个量。函数的定义通常分为传统定义和近代定义，函数的两个定义本质是相同的，只是叙述概念的出发点不同，传统定义是从运动变化的观点出发，而近代定义是从集合、映射的观点出发。

1. 传统定义

一般，在一个变化过程中，有两个变量 x、y，如果给定一个 x 值，相应的可以确定唯一的一个 y，那么就称 y 是 x 的函数。其中 x 是自变量，取值范围叫做这个函数的定义域；y 是因变量，y 的取值范围叫做函数的值域。

2. 近代定义

设 A，B 是非空的数集，如果按照某种确定的对应关系 f，使对于集合 A 中的任意一个数 x，在集合 B 中都有唯一确定的数 $f(x)$ 和它对应，那么就称 $f(A)\rightarrow B$ 为从集合 A 到集合 B 的一个函数，记作 $y=f(x)$，$x\in A$。

其中，x 叫做自变量，x 的取值范围 A 叫做函数的定义域；与 x 值相对应的 y 值叫做函数值，函数值的集合 $\{f(x)\mid x\in A\}$ 叫做函数的值域。

定义域、值域和对应法则称为函数的三要素，一般书写为 $y=f(x)$，$x\in D$。若省略定义域，一般是指使函数有意义的集合。

3. 程序中的定义形式

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

其中，类型标识符处若不说明类型，一律自动按整数处理；形式参数是被初始化的内部变量，生命周期和可见性仅限于函数内部，若无参数，写 void。

3.3.4 表达式

表达式是由数字、算符、数字分组符号(括号)、自由变量和约束变量等能求得数值的有意义排列方法所得的组合。约束变量在表达式中已被指定数值，而自由变量则可以在表达式之外另行指定数值。

表达式表示程序要执行的一个操作过程，其结果是一个数据，即表达式的值。因此，可以将表达式看作一种数据表示手段，而这种手段需要经过运算。根据不同的运算类别，常用的表达式有算术表达式、关系表达式、逻辑表达式、赋值表达式、条件表达式等。

一个表达式必须是合式的，即每个算符都必须有正确的输入数量。如表达式 2+3 便是合式的；而表达式×2+则不是合式的，至少不是算术的一般标记方式。两个表达式若被说是等值的，表示对于自由变量任意的定值，两个表达式都会有相同的输出，即它们代表同一个函数。

3.3.5 数据类型

数据类型是对程序中被处理数据的抽象，所谓数据类型是按被说明量的性质、表示形式、占据存储空间的多少、构造特点来划分的。例如，C 语言规定在程序中使用的每个数据都属于一种数据类型。在 C 语言中，数据类型可分为基本数据类型、构造数据类型、指针类型、空类型四大类，如图 3.10 所示。

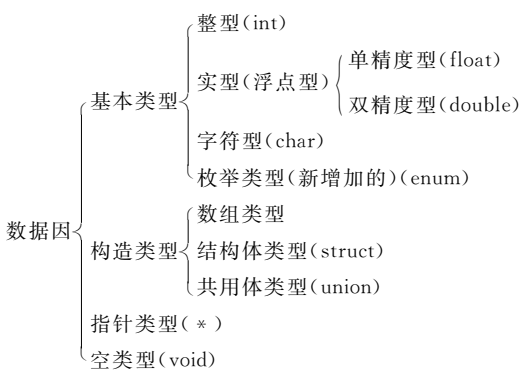

图 3.10 C 语言数据类型分类示意图

基本数据类型也称作简单数据类型。例如，Java 语言有 8 种简单数据类型，分别是 boolean、byte、short、int、long、float、double、char，这 8 种数据类型可分为下列 4 大类型。

- 逻辑类型：boolean；
- 字符类型：char；
- 整数类型：byte、short、int、long；
- 浮点类型：float、double。

3.4 机器层的数据表示

机器层的数据有数值、字符、图像、声音、文本、程序等多种，最终都会以某种编码方式转换成二进制形式。

3.4.1 数值型数据的表示

当数据仅为数值时，用二进制计数法对数值数据进行编码。

1. 数制及其转换

r 进制即 r 进位制。r 进制数 N_r 写为按权展开的多项式之和：

$$N_r = \sum_{i=-\infty}^{+\infty} D_i \times r^i \tag{3.1}$$

其中，D_i 是该数制采用的基本数符号，r^i 是第 i 位的权，r 是基数。例如，十进制数 123456.7 可以表示为：$123456.7 = 1\times10^5 + 2\times10^4 + 3\times10^3 + 4\times10^2 + 5\times10^1 + 6\times10^0 + 7\times10^{-1}$。

计算机中常用的计数制是二进制、八进制和十六进制。

(1) 二进制的运算法则

二进制加法的进位法则是“逢二进一”，即 0+0=0，1+0=1，0+1=1，1+1=10。二进制减法的进位法则是“借一为二”，即 0−0=0，1−0=1，1−1=0，10−1=1。二进制乘法规则：0×0=0，1×0=0，0×1=0，1×1=1。二进制除法是乘法的逆运算，类似十进制除法。

(2) 十进制与二进制相互转换

算法：将十进制整数部分除以 r 取余，小数部分乘以 r 取整，最后将两部分合并。

【例 3.4】 将十进制数$(347.625)_{10}$转化为二进制数。

解：

步骤一：转换整数部分

347/2 = 173……余1
173/2= 86……余1
86/2= 43……余0
43/2= 21……余1
21/2= 10……余1
10/2= 5……余0
5/2= 2……余1
2/2= 1……余0
1/2 = 0……余1

$(347)_{10} = (101011011)_2$

步骤二：转换小数部分

$$0.625\times2=1.25 \quad 1$$
$$0.25\times2=0.5 \quad 0$$
$$0.5\times2=1 \quad 1$$
$$(0.625)_{10}=(101)_2$$

得：

$$(347.625)_{10}=(101011011.101)_2$$

二进制、八进制、十进制和十六进制的对应关系如表 3.1 所示。

表 3.1 二进制、八进制、十进制和十六进制的对应关系

二进制	八进制	十进制	十六进制	二进制	八进制	十进制	十六进制
000	0	0	0	1000	10	8	8
001	1	1	1	1001	11	9	9
010	2	2	2	1010	12	10	A
011	3	3	3	1011	13	11	B
100	4	4	4	1100	14	12	C
101	5	5	5	1101	15	13	D
110	6	6	6	1110	16	14	E
111	7	7	7	1111	17	15	F

2. 机器数和码制

各种数据在计算机中的表示形式称为机器数，其特点是采用二进制数。计算机中表示数值数据时，为了便于运算，带符号数常采用原码、反码、补码三种编码方式，这种编码方式称为码制。为区别起见，将带符号位的机器数对应的真正数值称为机器数的真值。

(1) 原码表示法

原码是一种计算机中对数字的二进制定点表示方法。原码表示法在数值前面增加了一位符号位(即最高位为符号位)：正数该位为 0，负数该位为 1(0 有两种表示：+0 和−0)，其余位表示数值的大小。例如，用 8 位二进制表示一个数，+1 的原码为00000001，−1 的原码为10000001。原码是人脑最容易理解和计算的表示方式。

原码不能直接参加运算，可能会出错。例如数学上，1+(−1)=0，而在二进制中 00000001+10000001=10000010，换算成十进制为−2，显然出错了。所以原码的符号位不能直接参与运算，必须和其他位分开，这就增加了硬件的开销和复杂性。

(2) 反码表示法

反码表示法规定：正数的反码与其原码相同；负数的反码是对其原码逐位取反，但符号位除外。例如，用 8 位二进制表示一个数，+1 的反码为00000001，−1 的反码为11111110。可见如果一个反码表示的是负数，人脑无法直观地看出它的数值，通常要将其转换成原码再计算。

(3) 补码表示法

补码表示法规定：正数的补码与其原码相同；负数的补码是在其反码的末位加 1。例如，用 8 位二进制表示一个数，+1 的补码为 00000001，−1 的补码为 11111111。对于负数的补码表示方式，人脑也是无法直观地看出其数值的，通常也需要转换成原码再计算。补码

表示法的一个主要优点是任何带符号数字组合的加法都可以利用相同的算法，也就可以用相同的电路，因此，应用补码表示法的计算机只需知道加法就可以了。今天计算机表示整数最普遍的系统就是补码计数法。

(4) 移码表示法

移码(又叫增码)是符号位取反的补码，一般用做浮点数的阶码，引入的目的是为了保证浮点数的机器零为全0。

3. 定点数和浮点数

计算机在处理数值数据时，对小数点的处理有两种不同的方法，分别是定点数据表示法和浮点数据表示法这两种不同形式的数据表示方法。

(1) 定点数

所谓定点数，就是小数点的位置固定不变的数。小数点的位置通常有两种约定方式：定点整数——纯整数，小数点在最低的有效数值位之后；定点小数——纯小数，小数点在最高有效数值位之前。表3.2是机器数字长为 n 时，原码、反码、补码、移码的定点数所表示的范围。

表3.2 机器数字长为 n 时表示的带符号的范围

码 制	定点整数	定点小数
原码	$-(2^{n-1}-1)\sim+(2^{n-1}-1)$	$-(1-2^{-(n-1)}-1)\sim+(1-2^{-(n-1)})$
反码	$-(2^{n-1}-1)\sim+(2^{n-1}-1)$	$-(1-2^{-(n-1)}-1)\sim+(1-2^{-(n-1)})$
补码	$-2^{n-1}\sim+(2^{n-1}-1)$	$-1\sim+(1-2^{-(n-1)})$
移码	$-2^{n-1}\sim+(2^{n-1}-1)$	$-1\sim+(1-2^{-(n-1)})$

(2) 浮点数

当机器字长为 n 时，定点数的补码和移码可以表示 2^n 个数，而其原码和反码只能表示 2^n-1 个数(正负0占了两个编码)。定点数所能表示的数值范围比较小，容易溢出，所以引入了浮点数。浮点数是小数点位置不固定的数，它能表示更大的范围，是一种基于科学计数法的计数表示法。

二进制数 N 的浮点数表示方法为：

$$N = 2^E \times F \tag{3.2}$$

其中，E 称为阶码，F 称为尾数。

在浮点表示法中，阶码通常为带符号的纯整数，尾数为带符号的纯小数。浮点数的一般表示格式如下。

阶码符号	阶码	尾数符号	尾数

浮点数的表示不是唯一的，当小数点的位置改变时，阶码也随之相应改变，因此可以用多种浮点形式表示同一个数。

3.4.2 非数值型数据的表示

非数值型数据主要有字符、汉字、图像、声音和视频等类型，每一种类型都可以通过二进

制编码方式进行编码。

1. 字符编码

在计算机中，除数字外，还需处理各种字符，如字母、运算符号、标点符号等等。计算机采用多种类型的编码来表示字符，就是用一个约定的二进制数来表示一个字符，主要的编码标准有 ASCII 码、EBCDIC 码和 Unicode 码。

(1) ASCII 码

ASCII 码(American Standard Code for Information Interchange)即信息交换美国标准码，由美国国家标准学会推出，该编码后来被国际标准化组织(International Organization for Standardization，ISO)采纳而成为一种国际通用的信息交换标准代码，即国际 5 号码。ASCII 码的长度为 7，一共有 2^7 种编码，从 0000000 到 1111111 可以表示 128 个不同的字符。这 128 个字符可以分为两类：可显示/打印字符 95 个和控制字符 33 个。可显示/打印字符包括 0～9 十个数字符，a～z、A～Z 共 52 个英文字母符号，“+”、“−”、“≠”、“/”等运算符号，“。”、“?”、“，”、“；”等标点符号，“♯”、“%”等商用符号在内的 95 个可以通过键盘直接输入的符号，它们都能在屏幕上显示或通过打印机打印出来。控制字符是用来实现数据通信时的传输控制打印或显示时的格式控制，以及对外部设备的操作控制等特殊功能，共有 33 个，它们都是不可直接显示或打印(即不可见)的字符。如编码为 7DH(最后一个字母 H 表示前面的 7D 用十六进制表示)的 DEL 用作删除操作，编码为 07H 的 BEL 用作响铃控制等。

扩展 ASCII 码有 8 位，这个字节全部用来表示字符，因此可表示 256 种符号和字母，其中前 128 种与常规 ASCII 码相同。

(2) EBCDIC 码

EBCDIC(Extended Binary Coded Decimal Interchange Code)即扩展的二/十进制交换码，由 IBM 公司发明，只用在旧式的 IBM 大型计算机上，并未在其他计算机中得到普及。EBCDIC 码采用 8 位表示一个字符，总共可以表示 2^8(256)个不同符号，但 EBCDIC 中并没有使用全部编码，只选用了其中一部分，剩下的保留用作扩充。在 EBCDIC 码制中，数字 0～9 的高 4 位编码都是 1111，而低 4 位编码则依次为 0000 到 1001。把高 4 位屏蔽掉，也很容易实现从 EBCDIC 码到二进制数字值的转换。

(3) Unicode 码

ISO 扩展的 ASCII 标准在支持全世界多语通信方面取得了巨大进展，但是仍有两个主要障碍。首先，扩展的 ASCII 码中额外可用的位模式数不足以容纳许多亚洲语言和一些东欧语言的字母表；其次，因为一个特定文档只能在一个选定的标准中使用符号，所以无法支持包含不同语种的语言文本的文档。实践证明，这两者都会严重妨碍其国际化使用。为弥补这一不足，Unicode 码在一些主要软硬件厂商的合作下诞生了，并迅速赢得了计算机行业的支持。

Unicode 为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求，1990 年开始研发，1994 年正式公布。国际组织制定的 Unicode 标准，采用两个字节来表示一个数字、字母、符号或文字，并为中文、日文等都分配了相应的码段(码值连续的区间)，以实现各种文字的国际交流。

2. 汉字编码

汉字的内部编码(内码)是在计算机处理汉字信息时所采用的机内代码,与汉字的输入编码不同,通常把汉字的输入码称为外部编码(外码),汉字还需要通过输出编码将其显示在屏幕上或用打印机打印出来。汉字内部编码以连续两个字节(16 位)来表示。汉字数远比 256 种多,不能占用 ASCII 码已经使用的码值。为了和英文字符的机内编码(ASCII 码)相区别,表示汉字的两个字节的最高位均置 1,这样两字节内码就可以表示 $2^{8-1}\times2^{8-1}$(16 384)个汉字。汉字外码的编码方法种类繁多,曾经被形容为万"码"奔腾,但主要可以分为数字编码、拼音码和字形码 3 类。

数字编码的特点是一字一码,无重码、编码长,且易和内部编码进行转换,但记忆各个汉字的编码是一件极其艰巨的任务,非专业人员很难使用。数字编码中,每一个汉字都分配给一个唯一的数字代码,用以代表该汉字,国际区位码、电报码都属于该类,常用的是国际区位码(又简称国际码或区位码)。国际区位码把 GB 2312 基本集中的 6737 个汉字分为 94 个区,每个区又分 94 位,以区码和位码的二维坐标形式给每个汉字进行编码。区码和位码各有两个十进制数字,每次输入一个汉字需击键 4 次。在 94 个分区中,1～15 区用来表示字母、数字和符号,16～87 区用以表示一级、二级汉字,其中一级汉字以汉语拼音为序排列,二级汉字以偏旁部首为序进行排列。

拼音码用每个汉字的汉语拼音符号作为汉字的输入编码。这种编码很容易学会使用,无须额外记忆,使用人员的负担小,因此成为最常用的一种方法,但是由于汉字同音字太多,重码率高,所以输入速度很难提高。

字形码以汉字的形状特点为每个汉字进行编码。最受欢迎的一种字形编码方法是五笔字型编码,是依据汉字的笔画特征将基本笔画分为点、横、竖、撇、折 5 类并分别赋以代号,另外根据汉字的结构特征把汉字分为上下型、左右型、包围型和单体型 4 种字型并分别赋以代号。汉字的五笔字型编码就是依据其组成部件和结构特征进行编码,其输入能达到很高的速度。

汉字的编码规则有多个,在我国大陆采用 GB 2312"信息交换汉字编码字符集-基本集",收集了常用汉字 6763 个:一级汉字 3755 个,二级汉字 3008 个。GB 18030 是 2000 年公布的"信息交换汉字编码字符集基本集扩充",收录 27 000 多汉字。2005 年公布最新版,收录 70 000 多汉字,包括少数民族文字。

3. 图像编码

数字是离散的对象,因此要处理图像,首先要将图像离散化。图像格式大致可以分为两大类:一类为位图,另一类为矢量图。位图把每幅图像看作是点的集合,每个点叫做像素,如黑白图像用一位数字表示一个像素点,0 表示亮点,1 表示暗点。像素的色彩通常根据三原色原理表示,即任何颜色都可以用由不同浓度的红、绿、蓝三色混合而成,每个像素的颜色用 24 个位(3 个字节)来表示。若一个图像用 1024×1024 个像素来表示,则存储这一图像大约用几兆字节,同时也不利于放大。解决位图容量大的问题可以使用图像压缩技术,如采用 GIF、JPEG 格式保存图像,可分别压缩 2/3、1/20。目前 JPEG 格式是彩色图像公认的有效标准,被众多相机厂商所接纳。矢量图中一幅图像不再是像素点的集合,而是一组直线和

曲线的集合，用数学方法来表示，它们是指示图像设备产生图形的依据，而不是图像的像素模式。这种表示方法适用于表示字符、汉字和线条图形，不适用于表示相片和图画等。

位图是以点阵即像素形式描述图像的，矢量图是以数学方法描述的由几何元素组成的图像。一般说来，位图由离散的像素组成，很难表示动态图像，如有时视频出现马赛克，就是缺少像素信息造成；矢量图对图像的表达细致、真实，缩小后图像的分辨率不变，在专业级的图像处理中运用较多。

图像的主要指标为分辨率、色彩数与灰度。分辨率一般有屏幕分辨率和输出分辨率两种，前者用每英寸行数与列数表示，数值越大，图像质量越好；后者衡量输出设备的精度，以每英寸的像素点数表示，数值越大越好。常见的色彩位表示一般有 2 位、4 位、8 位、16 位、24 位、32 位、64 位这几种。图像若是 16 位图像，即为 2 的 16 次方，共可表现 65 536 种颜色；当图像达到 24 位时，可表现 1677 万种颜色，即真彩。

4. 声音编码

模拟音频信号是一种时间上连续的数据，数字编码的声音是一个数据序列，在时间上只能是间断的，因此当把模拟声音变成数字声音时，需要每隔一个时间间隔对音频信号幅度进行测量，称为采样，该时间间隔为采样周期(其倒数为采样频率)。采样之后的数据再量化，用二进制编码表示出来。由此看出，数字声音是经过采样、量化和编码后得到的，数字化声音的过程如图 3.11 所示。

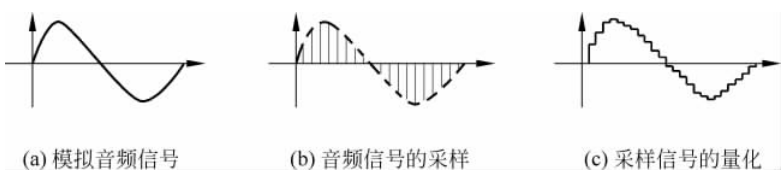

图 3.11　声音数字化示意图

为了保证声音还原的音质，采样频率要足够高。采样频率通常有三种：11.025kHz(语音效果)、22.05kHz(音乐效果)、44.1kHz(高保真效果)，常见的 CD 唱盘的采样频率即为 44.1kHz。

显然，采样频率越高、编码位数越多，声音的失真程度就越小。通常用 16 或 32 位来对声音振幅的测量值进行编码。

习题 3

1. 列举数据表示的层次，并解释为什么要分层次地表示数据？

2. 各种数据在计算机内的共同表示特点是什么？

3. 设某商业集团的仓库管理系统数据库有三个实体集。一是“公司”实体集，属性有公司编号、公司名、地址等；二是“仓库”实体集，属性有仓库编号、仓库名、地址等；三是“职工”实体集，属性有职工编号、姓名、性别等。

公司与仓库间存在“隶属”联系，每个公司管辖若干仓库，每个仓库只能属于一个公司管

辖；仓库与职工间存在“聘用”联系，每个仓库可聘用多个职工，每个职工只能在一个仓库工作，仓库聘用职工有聘期和工资。

试画出 ER 图，并在图上注明属性、联系的类型。

4. 数据结构和数据类型两个概念之间有区别吗？如果有，区别是什么？

5. 对数据的逻辑结构和存储结构简要介绍，并说明它们的分类。

6. 举出现实世界数据对象例子，它们分别具有下列数据结构：线性表、栈、队列、树、图。

7. 列举高级语言层表示数据的 5 种基本手段。

8. 举例说明术语：数组、数组类型、数组元素、数组元素个数和数组下标的概念。

9. 简述函数机制的 3 个要点。

10. 机器数 0011 和 1101 代表两个有符号的整数，用原码规则解释，它们的真值是多少？用补码规则解释，它们的真值是多少？

11. 补码的符号位和原码的符号位的根本区别是什么？

12. 字符编码的常用标准有哪些？并做简单介绍。

13. 已知英文字母“a”的 ASCII 码是 1100001，那么英文单词“beef”的二进制编码是什么？

14. 设彩色图像每个像素用 3 个字节表示，一幅图像有 1024×1024 个像素，计算需要的存储容量。