第 1 章　绪　　论

本章主要内容

本章简单介绍离散数学研究的对象以及在信息技术中的应用，主要内容包含连续量和离散量的定义，以图像数字化和语音数字化为例说明了连续量的离散化过程；以数据库理论、编译系统、人工智能、密码学和程序语义为例简单介绍了离散数学在信息技术方面的应用。

1.1　离散数学的研究对象

在现实生活和生产中会产生各种数据，它们的性质各不相同，有的数据在时间上和数量上是连续变化的，在坐标平面上的图像表现为一条连续的曲线，这种量称为连续量，物理学上把连续量称为模拟量。例如，交流电的电压就是一个周期性连续变化的量，是一个连续量；一个人身高的变化也是一个连续量等。

还有一些量是分散开来的，不存在中间值，也就是说它的变化在时间上是不连续的，总是发生在一系列离散的瞬间，这样的量称为离散量。自然界的很多量都是离散的，例如，某个时刻在你视野里的桌椅板凳的数量等，其变化都是整数之间的跳变。以上这些例子的数值是有限的，但也可能有无限个数值的情况，如整数是离散量，但它是无限的。

信息技术的发展使得计算机已经深入到了千家万户，并大大改变了人们的生活方式。计算机只能识别 0 和 1 两个数字，也就是它只能处理离散数据。为了在计算机中表示离散数据，需要对离散数据进行编码处理，得到的量称为数字化量。

用计算机处理某个问题时，首先要用计算机能识别的方式把问题描述出来。对于离散量来说，就是要把它们转化为数字量；对于连续量来说，首先需要离散化，转化为离散量，再转化为数字量。不仅如此，还需要把离散量之间的关系描述清楚。这个研究离散量的结构及其相互关系的学科称为离散数学。由于数字电子计算机只能处理离散的或离散化了的数量关系，因此，在数字电子计算机问世之后离散数学学科的研究显得尤为重要，因为无论计算机科学本身，还是与计算机科学及其应用密切相关的现代科学研究领域，都面临着如何对离散结构建立相应的数学模型的问题，以及如何把已经用连续数量关系建立起来的数学模型离散化的问题。离散数学课程包含以下 6 个方面的内容。

① 集合论部分：集合及其运算、二元关系与函数及其应用。

② 图论部分：图的基本概念、图的矩阵表示、特殊图、树、平面图及其应用。

③ 代数结构部分：代数系统的基本概念、半群与独异点、群、环与域、格与布尔代数及其应用。

④ 数理逻辑部分：命题逻辑、一阶谓词演算、消解原理及其应用。

⑤ 形式语言与自动机理论：语言的表示与生成、语言的识别及其在编译系统中的应用。

⑥ 组合数学部分：递推和递归、组合存在性定理、基本的计数公式、组合计数方法、

组合计数定理和组合设计等。

离散数学课程中所包含的思想和方法被广泛地应用于信息技术相关专业的诸多领域，从科学计算到信息处理，从理论计算机科学到计算机应用技术，从计算机硬件到计算机软件，从人工智能到认知系统和大数据，无不与离散数学密切相关。

1.2 连续量的数字化

这一节以图像的数字化和语音的数字化为例说明连续量的数字化过程。

1.2.1 图像的数字化

图像在计算机中的表示方式有两种，一种是用像素点阵方法记录，即位图；另一种是通过数学方法记录图像，即矢量图。这里以位图表示法为例进行说明。

像素是指构成图像的小方块，这些小方块都有一个明确的位置和被分配的色彩数值，小方格颜色和位置就决定该图像所呈现出来的样子。可以将像素视为整个图像中不可分割的单位或者是元素。不可分割的意思是它不能够再切割成更小单位，是一个单一颜色的小格。每一个点阵图像包含了一定量的像素，这些像素决定图像在屏幕上所呈现的大小。这样，图像是由一个个像素组成的，并且赋予像素一个表示颜色的值，可以用数组存储这些像素。如果是黑白图像，那就可以用 1 位二进制数表示即可，也就是 0 代表白，1 代表黑。对于彩色图像，可以用一个数来表示其颜色。根据三基色原理，利用 R（红）、G（绿）、B（蓝）三色不同比例的混合来表现丰富多彩的现实世界，每个颜色的深度可以使用 0～255 这 256 个数字来表达，例如（255,255,255）就可以表示白色，（0,0,0）就可以表示黑色。如果规定了图片的长和宽的比例，如 3 像素×3 像素的图片，该图像的表示方式如图 1-1 所示。

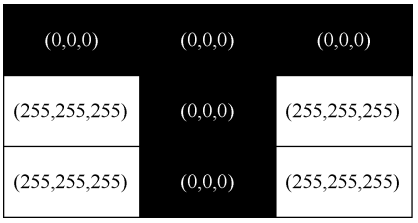


图 1-1 3 像素×3 像素的图片示例

将每一个格子的数据按照从左到右、从上到下依次写下来，便可以存储图片的信息。存储位图的文件通常包含多种信息，如每个像素的颜色信息、行数和列数等。另外，此类文件可能还包含颜色表，也称为颜色调色板。颜色表将位图中的数值映射到特定的颜色。

将模拟图像数字化的步骤包含采样和量化两个过程。

采样：将空间上连续的图像变换成离散点的操作称为采样。简单地讲，对二维空间上连续的图像在水平方向和垂直方向上等间距地分割成矩形网状结构，所形成的微小方格称为像素点。

量化：将像素灰度转换成离散的整数值的过程叫量化。采样后图像被分割成空间上离散的像素，但其灰度是连续的，还不能用计算机进行处理。必须把它们的灰度也离散化处理后，才能进行计算机处理。

数字化后得到的图像数据量十分巨大，必须采用编码技术来压缩其信息量。从一定意义上讲，编码压缩技术是实现图像传输与储存的关键。

1.2.2 语音的数字化

声音是由物体振动产生的，是一种波。例如，人们面对话筒说话时，声波通过话筒转变

为时间上连续的电压波，电压波与引起电压波的声波的变化规律是一致的，因此，可以利用电压波来模拟声音信号，这种电压波被称为模拟音频信号，如磁带、录像带上的声音信号，其波形图如图 1-2 所示。播放时音响设备将电压波传至扬声器，扬声器的振动产生声音，从而将模拟音频电信号还原为声音。

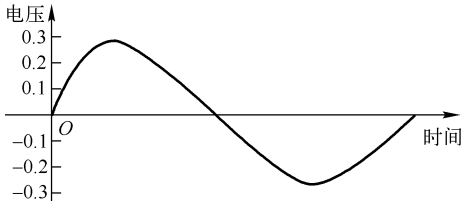


图 1-2　声波示意图

把模拟量的语音信号转化为数字量的语音信号需要经过采样、量化、编码三个过程。

（1）采样

对连续信号按一定的时间间隔采样。奈奎斯特采样定理认为，只要采样频率大于等于信号中所包含的最高频率的两倍，则可以根据其采样完全恢复出原始信号，这相当于当信号是最高频率时，每一周期至少要采取两个点。但这只是理论上的定理，在实际操作中，人们用混叠波形，从而使取得的信号更接近原始信号。把分割线与信号图形交叉处的坐标位置记录下来，就得到了所需要的数据。例如，假设采样时间间隔是固定的 0.01 s，则得到（0.01，0.10），（0.02,0.20），（0.03,0.26），（0.04,0.30），…，这样就把这个波形以数字记录下来了，如图 1-3 所示。事实上，只要把纵坐标记录下来就可以了，得到的结果是 0.10，0.20，0.26，0.30，…。

（2）量化

采样的离散音频要转化为计算机能够表示的数据范围，这个过程称为量化。量化的等级取决于量化精度，也就是用多少位二进制数来表示一个音频数据。一般有 8 位、12 位或 16 位。量化精度越高，声音的保真度越高。

（3）编码

对音频信号采样并量化成二进制，实际上就是对音频信号进行编码，但用不同的采样频率和不同的量化位数记录声音，在单位时间中，所需存储空间是不一样的。波形声音的主要参数包括采样频率、量化位数、声道数、压缩编码方案和码率等。未压缩前，波形声音的码率计算公式为：波形声音的码率=采样频率×量化位数×声道数/8。量化和编码后的图像示例如图 1-4 所示。采样电压、量化和编码的示例如表 1-1 所示。波形声音的码率一般比较大，所以必须对转换后的数据进行压缩。

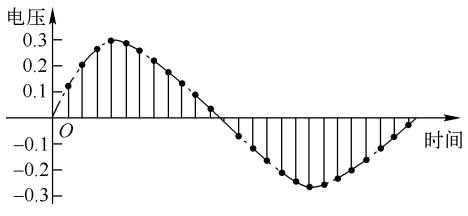


图 1-3　采样后的图形

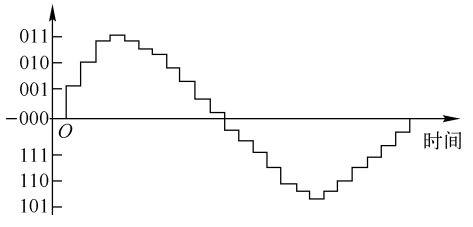


图 1-4　量化和编码后的图形

表 1-1 采样电压、量化和编码的示例

采 样 电 压	量化（十进制数）	编码（二进制数）
0.5~0.7	3	011
0.3~0.5	2	010
0.1~0.3	1	001
-0.1~0.1	0	000
-0.3~-0.1	-1	111
-0.5~-0.3	-2	110
-0.7~-0.5	-3	101
-0.9~-0.7	-4	100

1.3 离散数学在信息技术中的应用

离散数学在信息技术中的应用非常广泛，这里列举部分实例作简要介绍。

1.3.1 离散数学与关系数据库

日常生活中，经常用到如图 1-5 所示的表格，如果是单张不复杂的表格，可以直接用 Excel 表处理就行，甚至数据之间不存在关联的表格，都可以方便地用 Excel 表处理，但对于数据之间相互关联的表格，如何处理呢？

学号	姓名	计算机导论	离散数学	数据结构	操作系统	编译原理
00001	张三	89	65	75	86	85
00002	李四	98	93	67	68	87
00003	王五	77	87	83	91	66

（a）

学号	姓名	课程名称
00001	张三	离散数学
00002	李四	数据结构
00003	王五	操作系统
00001	张三	编译原理
00003	王五	离散数学
00002	李四	离散数学
00001	张三	数据结构

（b）

职工号	姓名	课程名称
12001	钱三	离散数学
21002	童四	数据结构
14003	栗五	操作系统
12001	钱三	编译原理
14003	栗五	离散数学
21002	童四	离散数学
12001	钱三	数据结构

（c）

图 1-5 学生成绩信息和选课信息以及教师的任课信息表

例如，图 1-5（b）是一个学生选课表，图 1-5（c）是一个教师任课表，那么如何通过这两张表找到选某老师的所有学生呢？用 Excel 表实现这个功能就比较困难了，这就涉及了数据的管理问题。计算技术的初期，随着计算机应用的普及和深入，数据的管理问题显得越来越重要，人们为了解决这个问题做了大量的工作，提出了一些解决的途径和模型。这其中的第一个问题就是数据的描述方法。因为用计算机处理问题，首先需要把该问题用计算机能识别的方式描述出来，这种描述问题的方式在计算机科学领域称为形式化方法。在人们提出的模型中，有三种方法得到了实现和使用，它们分别是层次模型、网络模型和关系模型，其

中的关系数据模型被广泛采用，因为它有着完备的理论基础和实现方法。

事实上，一个具有 n 个属性的二维表格是由一条一条的信息组成的，一条信息也称为记录，一组记录就构成了一个二维表。图 1-5（a）的表格，可以以班级为单位，每个班级有一个表格，可以把一个专业的几个班的表格集中起来形成一个专业的成绩单；同样，可以把各个专业的表格集中起来形成全校学生的成绩单。从形式化的角度看，或者说从数学的角度看，数据之间的关系可以用集合理论中的关系来表述，一条记录也就是 n 条数据的一个 n 元组，一个二维表也就是一个 n 元组的集合，即一个 n 元关系，也就是说，一个表格就是一个 n 元关系；表格的合并就相当于关系的并运算；找两个表的相同元组相当于求两个关系的交集。以此类推，数据集上的一些其他操作也都对应着关系中的某个操作。

科德（Codd）对这种思想和方法进行了深入研究，于 1970 年首次提出了数据库系统的关系模型，也就是数据库就是一个 n 元关系，可以存放在计算机中的一个具有 m 行和 n 列的二维数组中，其中每一行的分量组成一个 n 元组，它是一条记录，代表一个完整的数据，它的分量称为记录的域；对应的实体可以有 m 条记录（m 个数据）；这就是数据库的关系模型，以关系模型为基础建立的数据库系统被称为关系数据库，也就是说，所谓关系数据库系统是指支持关系模型的数据库系统。

用户使用关系数据库就是对一些二维数组进行检索、插入、修改和删除等操作。为此数据库管理系统必须向用户提供使用数据库的语言，即数据子语言。这种语言目前是以关系代数或谓词逻辑为其数学基础。由于引入了数学方法，使得关系数据库比其他几种数据库更具有优越性，从而得到了迅猛的发展，目前已替代了其他类型的数据库，并成为数据库中最有实用价值和理论价值的数据模型。当今流行的各种大型网络数据库都支持关系模型，如 Oracle、SQL server 等。

关系模型由关系数据结构、关系操作集合和关系完整性约束三部分组成。

（1）关系模型的关系数据结构

在关系模型中，现实世界的实体以及实体间的各种联系均用关系来表示，所以其数据结构非常单一。在用户看来，关系模型中数据的逻辑结构是一张二维数据表。

（2）关系模型的关系操作集合

关系模型只给出了关系操作的能力的描述，但没有对关系数据库系统语言给出具体的语法要求。

关系模型中常用的操作包括并（union）、交（intersection）、差（difference）、选择（select）、投影（project）、连接（join）、除（divide）等查询（query）操作和插入（insert）、删除（delete）、修改（update）操作两大部分。查询的表达能力是其中最主要的部分。

早期的关系操作能力通常用代数方式或逻辑方式来表示，分别称为关系代数和关系演算。关系代数是用对关系的运算来表达查询要求的方式；关系演算是用谓词来表达查询要求的方式。关系演算又可按谓词变元的基本对象是元组变量还是域变量分为元组关系演算和域关系演算。关系代数、元组关系演算和域关系演算三种语言的表达能力是完全等价的。关系语言是一种高度非过程化的语言，用户不必请求数据库管理员为其建立特殊的存取路径，存取路径的选择由数据库管理系统的优化机制来完成。此外，用户不必求助于循环结构就可以完成数据操作。

关系代数、元组关系演算和域关系演算均是抽象的查询语言，与具体的数据库管理系统

中实现的实际语言并不完全一样，是评估实际系统中查询语言能力的标准或基础。实际的查询语言除了提供关系代数或关系演算的功能外，还提供了许多附加功能，如集函数、关系赋值和算术运算等。

(3) 关系模型的关系完整性约束

关系模型允许定义三类关系完整性约束，分别是实体完整性、参照完整性和用户定义的完整性。其中实体完整性和参照完整性是关系模型必须满足的关系完整性约束条件，体现了具体领域中的语义约束。

① 实体完整性。实体完整性是针对基本关系而言的。一个基本表通常对应现实世界的一个实体集，如学生关系对应于学生的集合。现实世界中的实体是可区分的，即它们具有某种唯一性标志，相应地，关系模型中以主码作为唯一性标志。主码中的属性即主属性不能取空值。所谓空值就是“不知道”或“无意义”的值。如果主属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体，因此，这个规则称为实体完整性。

② 参照完整性。若属性（或属性组）F 是基本关系 R 的外码，它与基本关系 S 的主码 K 相对应（基本关系 R 和 S 不一定是不同的关系），则对于 R 中的每个元组在 F 上的值必须为（或者取）空值（F 的每个属性值均为空值），或者等于 S 中某个元组的主码值。

③ 用户自定义的完整性。用户定义的完整性就是针对某一具体关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求。例如，某个属性必须取唯一值、某些属性值之间应满足一定的函数关系、某个属性的取值范围在 0~100 之间等。关系模型应提供定义和检验这类完整性的机制，以便于用统一的系统的方法处理它们，而不是由应用程序承担这一功能。

1.3.2 形式语言与编译系统

所谓程序设计语言就是计算机所能识别的语言。程序设计语言由命令的集合构成，通过这些命令构成的序列得到求解问题的过程，这就是程序，也就是说，计算机程序就是用计算机语言书写的、能完成一定功能的代码序列。随着计算机技术的发展，用于程序设计的计算机语言也不断地向语言更加丰富、语句更容易理解的方向发展，以扩大计算机的应用范围。同样，计算机程序设计语言也经历了从机器语言到汇编语言，再到高级程序设计语言的发展历程。由于计算机只识别 0、1 代码，高级程序语言写成的语言需要翻译为 0、1 代码的机器语言，这个工作就是由编译系统完成的，编译系统的理论基础就是形式语言及自动机理论。

利用高级语言编写程序的过程是：首先借助编辑软件系统编辑得到高级语言源程序；利用高级语言的翻译程序将高级语言源程序自动翻译成目标程序；再将目标程序通过连接程序自动生成可执行文件。整个过程如图 1-6 所示。

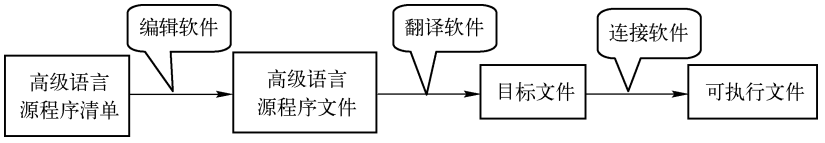


图 1-6 高级语言程序的执行过程

随着计算机软件的发展，出现了集成化环境。所谓集成化环境就是将程序的编辑、编译或解释、连接、运行等操作集成在一个环境中，把各种命令设计成菜单命令。这样，更加方便了

非计算机专业人员掌握利用高级语言设计程序的过程。在集成环境中除了程序的主要操作命令外，还设计了文件操作的命令，如打开、保存、关闭等，以及程序调试命令、分步操作、跟踪、环境设置等，方便程序员在集成环境下进行程序的编写、调试、运行。例如，在 VC++6.0 的集成环境中，编写一个求 1~100 的整数的和，并输出结果的程序，如图 1-7 所示。

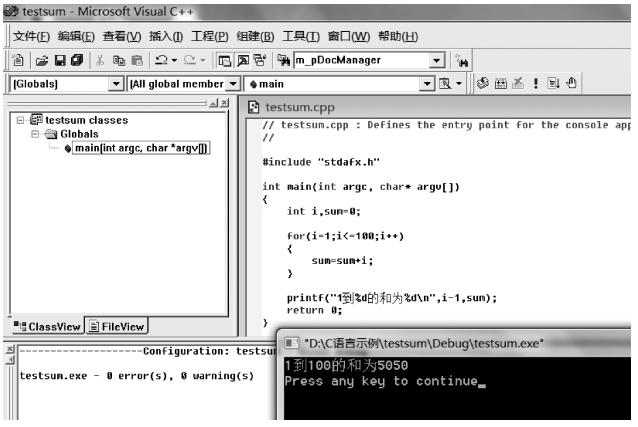


图 1-7　编译系统的执行示例

在上述程序中，如果没有声明变量 i，则会提示错误，如图 1-8 所示。在这个过程中，如果程序正确，则得到一个可执行的 exe 文件，运行该文件就输出执行结果。如果程序中存在错误，系统就会提示程序存在错误。系统是如何识别出错误的呢？其工作的原理和基础是什么？其工作原理的基础就是形式语言及自动机理论，这涉及两个问题。第一个问题就是如何生成一个语言。例如，C 语言规定标识符只能由字母、数字和下划线组成，并且以下划线或者字母开头。那么如何生成 C 语言的标识符呢？这就涉及了形式语言中的所谓的文法，可以定义如下的文法来生成 C 语言的标识符。

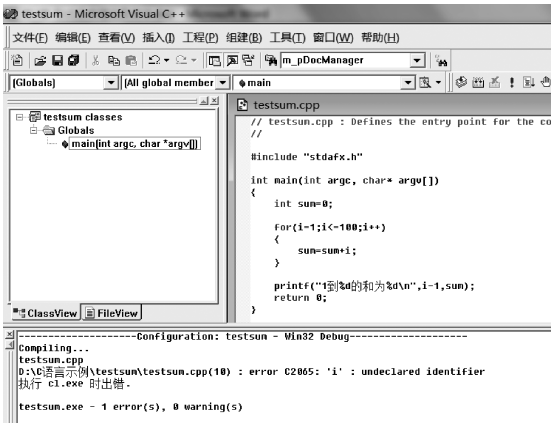


图 1-8　编译系统的错误提示示例

<LABEL>→<LABEL><LETTER> | <LABEL><DIGITAL> | <LABEL>_　　(1)

<LABEL>→<LETTER> | _　　(2)

<LETTER>→a | b | c | … | z | A | B | C | … | Z　　(3)

<DIGITAL>→1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9　　(4)

<LABEL>用于表示标识符，<LETTER>表示字母。符号“→”最早来源于 Post 的产生式，表示推出的意思，在语言生成中表示替换。例如，式（1）表示在<LABEL>出现的地方可以用<LABEL><LETTER>、<LABEL><DIGITAL>或<LABEL>_替换，其中的符号“|”表示“或”的意思。如果用“_”替换<LABEL>，也就得到“_”，即得到了标识符“_”。如果用<LETTER>替换<LABEL>，再由式（3），若用 A 替换<LETTER>，则得到标识符“A”。一个文法生成的字符串的集合也称为文法生成的语言。

第二个问题就是如何识别一个字符串是否属于某个语言。这也是编译系统的第一步工作，称为词法分析，可以用状态转换图来描述语言的识别的过程。所谓状态图是一个有向图，其结点代表状态，用圆圈表示；状态之间用有向弧连接，弧上的标记代表在射出结点（即箭弧始结点）状态下可能出现的输入字符或字符类。并且，它有至少一个称为初始状态的结点，用→○表示；存在一些称为终止状态的结点，用◎表示。状态转换图中可以没有终止状态，但事实上，至少应该有一个终止状态。图 1-9 给出了一些状态转换图的例子，其中图 1-9（c）是识别 C 语言标识符的状态转换图。

根据图 1-9（c），可以很容易地编程实现识别一个字符串是否符合标识符的要求，如果输入的字符串是合式的标识符，则输出为 0，否则输出为 1。例如，输入的字符串为“fskfsk_fldg33”“_gkfdgd323n_fs”“_”等合法的字符串，该函数的返回值为 0。但对于“34fflsl”“gss+sfs2”等不合法的字符串，返回值为 1。甚至可以把错误的具体位置等信息反馈给编程者作为参考，这就是如图 1-8 所示的编译系统提示错误的情况。

编译器的输入是语言的源文件，一般是文本文件。对于输入的文件，首先要分离出这个输入文件的每个元素，如关键字、变量等。然后根据语言的文法，分析这些元素的组合是否合法，以及这些组合所表达的意思。程序设计语言的每个特定的符号表示特定的意思，而且程序设计语言是上下文无关的。上下文无关就是某一个特定语句所要表达的意思和它所处的上下文没有关系，只由它自身决定。

编译器的执行过程如图 1-10 所示，一般经历词法分析、语法分析、语义分析、中间代码生成和目标代码生成的过程，中间还有代码优化的过程。代码优化过程很重要，可以节省程序所占空间，提高程序执行效率。

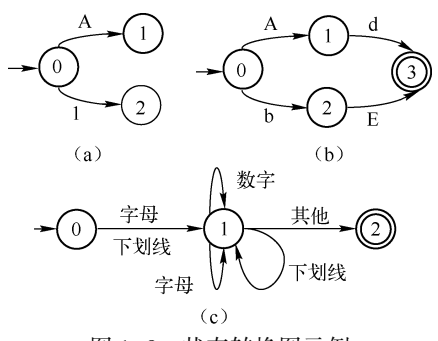


图 1-9　状态转换图示例

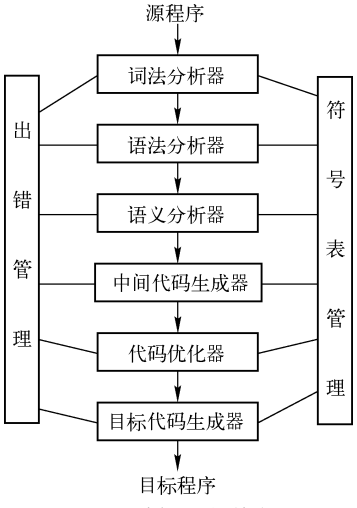


图 1-10　编译器的执行过程

1.3.3　数理逻辑与程序设计语言

在程序设计语言的家族中，典型的范型有过程式、面向对象、函数式和逻辑式。过程式程序设计语言的基本观点就是强制改变内存中的值，所以人们也称这种语言为命令式或者强制式，C 语言是其中的典型代表之一。面向对象语言的基本观点就是将数据和其上的操作封装于对象中，最显著的特点是封装、继承和多态，典型的语言有 Java、C++等。函数式语言的基本观点就是程序对象是函数及高阶函数，组织程序的范型是函数定义及引用，代表语言有 LISP、FP、ML、Miranda。逻辑式语言的基本观点就是程序对象是常量、变量和谓词，组织程序的范型是定义谓词，引用谓词的公式，并构造满足谓词的事实库和约束关系库，代表语言有 Prolog。

逻辑式程序设计语言基于自动定理证明理论，有其独特的程序设计风格，命令风格就是“A:-B1,B2,B3,…,Bn”，其意义就是：A 成立的前提条件是 B1,B2,B3,…,Bn 成立，所以要证明 A 成立，则需要证明 B1,B2,B3,…,Bn 成立；也可以理解为，若要求解 A，需要求解 B1,B2,B3,…,Bn。这样的语句称为规则，用数理逻辑的公式表示为“$B1 \wedge B2 \wedge B3 \wedge \cdots \wedge Bn \rightarrow A$”，子句的形式为“$\neg B1 \vee \neg B2 \vee \neg B3 \vee \cdots \vee \neg Bn \vee A$”。

对于某个 Bj，如果已知其成立或者知道其求解结果，不需要再进一步证明或者求解，把这样的语句称为事实，在语法上用“Bj.”表示。

对于不是事实的 Bi($1 \leqslant i \leqslant n$)还可以进一步细化，如“Bi:- C1,C2,C3,…,Cm”，意思就是“若要证明 Bi 成立，则需要证明 $C1,C2,C3,\cdots,Cm$ 成立”，如此下去。

如上形式的一些规则和事实组成规则库。在规则库建立完后，就可以进行问题求解或者定理的证明了。对于要求的问题或者要证明的结论，用“?-D”的形式表示。

【例 1-1】 由如下①~⑥条的规则和事实构成规则库，⑦是问题。

① likes(bell,sports).

② likes(mary,music).

③ likes(mary,sports).

④ likes(jane,smith).

⑤ friend(john,X):-likes(X,reading),likes(X,music).

⑥ friend(john,X):-likes(X,sports),likes(X,music).

⑦ ?-friend(john,Y).

可以看出，这个程序中有四条事实、两条规则和一个问题，都是分行书写的。规则和事实可连续排列在一起，其顺序可随意安排，但同一谓词名的事实或规则必须集中排列在一起。问题不能与规则及事实排在一起，它作为程序的目标要么单独列出，要么在程序运行时临时给出。

这个程序的事实描述了一些对象（包括人和事物）间的关系；而规则描述了 john 交朋友的条件，即如果一个人喜欢读书并且喜欢音乐（或者喜欢运动和喜欢音乐），则这个人就是 john 的朋友，当然，这个规则也可看作是 john 朋友的定义；程序中的询问是“约翰的朋友是谁?”。

如上的四条事实和两条规则可以用逻辑公式表示为 likes(bell,sports)、likes(mary,music)、likes(mary,sports)、likes(mary,sports)、$\text{friend}(\text{john},X) \vee \neg \text{likes}(X,\text{reading}) \vee$

¬likes(X,music)和 friend(john,X) ∨¬likes(X,sports) ∨¬likes(X,music)。求解的过程也就是把询问否定了，这里就是¬friend(john,Y)，然后从目标的否定开始执行归结推理的过程，直到得到问题的答案或者不能再执行归结为止，对于该例来说答案就是“mary”。具体的求解过程，将在 3.4.2 节中讨论。

例 1-1 中只有一个 Prolog 程序中的目标，也可以含有多个语句。例如，对上面的程序，其问题也可以是“?-likes(bell,sports),likes(mary,music),friend(john,X).”等。如果有多个语句，则这些语句称为子目标。

1.3.4 代数系统与密码学

密码学从古至今都有着非常重要的作用，从以前的手动加密解密，到机械加密解密，再到现在的计算机加密和解密，经历了一个很长的过程，也出现了很多加密的方法。其中有一种方法，它不改变明文的符号，只是根据一定的规则把明文重新排列，以便打破明文的结构特性，这种密码称为置换密码，又称换位密码。

接下来的问题就是如何描述位置的变换？这里，可以用置换的形式来表示位置的变换。例如，对于一个有限集合 $S=\{1,2,\cdots,n\}$ 的置换 f，可以用如下的形式表示。

$$\begin{pmatrix} 1 & 2 & \cdots & n \\ f(1) & f(2) & \cdots & f(n) \end{pmatrix}$$

也就是说，第 1 个位置上的字符用 $f(1)$ 位置上的字符替换，第 2 个位置上的字符用 $f(2)$ 位置上的字符替换，以此类推，第 n 个位置上的字符用 $f(n)$ 位置上的字符替换，也就是说，所有的字符都没改变，只是位置改变了。

更具体地，对于置换 $f=\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}$ 就是把第 1 个位置的字符换到第 2 个位置上，把第 2 个位置上的字符换到第 3 个位置上，第 3 个位置上的字符换到第 1 个位置上，第 4 个位置上的字符保持不变。在做了这样的变换后，原来文字表达的信息就都变化了。

$$\begin{pmatrix} B & e & i & j \\ i & n & g & 2 \\ 0 & 0 & 8 & O \\ l & y & m & p \\ i & c & G & a \\ m & e & s & \end{pmatrix} \qquad \begin{pmatrix} i & B & e & j \\ g & i & n & 2 \\ 8 & 0 & 0 & O \\ m & l & y & p \\ G & i & c & a \\ s & m & e & \end{pmatrix}$$

（a）明文　　（b）密文

图 1-11　置换密码实例

例如，假设明文为“Beijing 2008 Olympic Games”，首先把明文中的空格去掉，明文变为“Beijing2008OlympicGames”，如果使用置换 f 进行加密，明文有 23 个字符，则需要再把明文分为一个 6×4 的矩阵，剩余不够的位置空缺即可，得到如图 1-11（a）中的矩阵。对矩阵中的每一行，按照 f 的定义进行变换，得到新的矩阵如图 1-11（b）所示，密文为“iBejgin2800OmlypGicasme”。

如果要把密文再还原为明文，则对密文应用逆置换就可以了。对置换 f 来说，逆置换为 $f^{-1}=\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}$，对密文在此应用该逆置换就可还原为原文。

对于这种方法，如果密码空间很小，很容易被破译，就是穷举密钥直到得到有意义的明文为止，最坏的情况就是把所有的置换都试一遍，就可以破译。例如，对于置换 f 来说，密钥空间只有 4!个元素，最坏也就是把这 24 种情况全部试一遍也就破译了密文。为了增加破译的难度，需要增加密钥空间。

增加密钥空间的途径有两种，一种是增加置换的维度，但英文字母只有 26 个，也就是

说如果仅仅是英文的内容，置换密码中密钥空间最大为 26!，破译的难度也不大。

另一种是在置换一次之后，再对新得到的密文进一步置换，甚至还可以再进一步地进行置换，这就大大地增加了密钥空间，增加了破译的难度。例如，对于英文来说，如果连续三次采用不同的置换进行加密，则密钥空间就变成了(26!)×(26!)×(26!)，这就是一个很大的数字了，也就是说密钥空间已经很大了，也就很难破译了。

对于这种方法，可以采用置换群来进行描述。设 G 是集合 S 上的所有置换的集合，○是置换的合成运算，则$<G,○>$就构成一个群，那么就可以用代数学的知识来研究密码。

置换密码最典型的一个例子就是转轮密码机。直到第一次世界大战结束，所有密码都是使用手工来编码的，就是笔加纸的方式。1918 年亚瑟·谢尔比乌斯（Arthur Scherbius）发明了转轮密码机，称为恩尼格玛（enigma），才彻底改变了手工加密的历史，实现了加密的机械化。图 1-12 是转轮密码机的基本组成示意图，转轮密码机由三个部分组成，分别是键盘、转子和显示器。

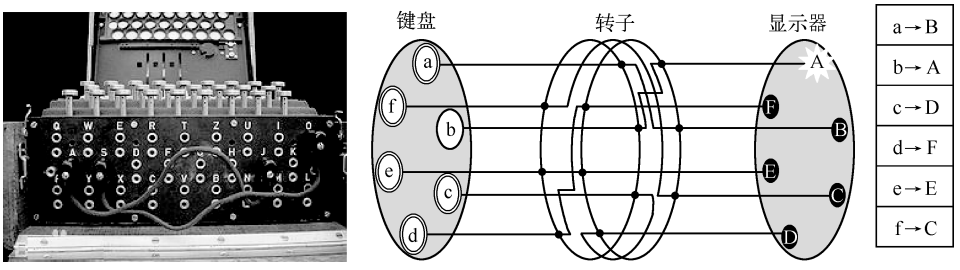


图 1-12　转轮密码机的基本组成示意图

恩尼格玛（enigma）上面共有 26 个键，键盘排列接近现在使用的计算机键盘。为了使消息尽量地短和更难以破译，空格和标点符号都被省略。在示意图中只画了 6 个键。实物照片中，键盘上方就是显示器，它由标示了同样字母的 26 个小灯组成，当键盘上的某个键被按下时，和此字母被加密后的密文相对应的小灯就在显示器上亮起来。同样地，在示意图上只画了 6 个小灯。在显示器的上方是 3 个转子，它们的主要部分隐藏在面板之下，在示意图中暂时只画了 1 个转子。

键盘、转子和显示器由电线相连，转子本身也集成了 6 条线路（在实物中是 26 条），把键盘的信号对应到显示器不同的小灯上去。在示意图中可以看到，如果按下 a 键，那么灯 B 就会亮，这意味着 a 被加密成了 B。同样，b 被加密成了 A，c 被加密成了 D，d 被加密成了 F，e 被加密成了 E，f 被加密成了 C。如果在键盘上依次键入 cafe（咖啡），显示器上就会依次显示 DBCE。这是最简单的加密方法之一，把每一个字母都按一一对应的方法替换为另一个字母，这样的加密方式叫作“简单替换密码”。

简单替换密码在历史上很早就出现了。著名的“恺撒法”就是一种简单替换法，它把每个字母和它在字母表中后若干个位置中的那个字母相对应。比如说取后三个位置，那么字母的一一对应就如下所示。

明文字母表：a b c d e f g h i j k l m n o p q r s t u v w x y z

密文字母表：D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

于是就可以从明文得到密文。假设明文为“veni，vidi，vici”，则密文为“YHQL，

YLGL，YLFL”。一般情况下这种对应关系比较随意，例如下面是另外一个例子。

明文字母表： a b c d e f g h i j k l m n o p q r s t u v w x y z

密文字母表： J Q K L Z N D O W E C P A H R B S M Y I T U G V X F

甚至可以自己定义一个密码字母图形而不采用字母。但是用这种方法所得到的密文还是相当容易被破译的。所以，如果转子的作用仅仅是把一个字母换成另一个字母，那就没有太大的意义了。但是大家可能已经猜到，所谓的“转子”，它会转动！这就是谢尔比乌斯关于 ENIGMA 的最重要的设计——当键盘上一个键被按下时，相应的密文在显示器上显示，然后转子的方向就自动地转动一个字母的位置，在示意图中就是转动 1/6 圈，而在实际中转动 1/26 圈。图 1-13 展示了连续键入 3 个 b 的情况。当第一次键入 b 时，信号通过转子中的连线，灯 A 亮起来，如图 1-13（a）所示。放开键后，转子转动一格，各字母所对应的密码就改变了，第二次键入 b 时，它所对应的字母就变成了 C，如图 1-13（b）所示。同样地，第三次键入 b 时，灯 E 闪亮，如图 1-13（c）所示。

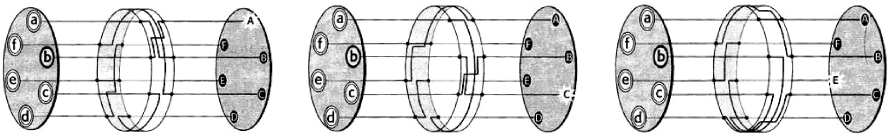


图 1-13 转轮密码机的工作过程示意图

这里作为一个应用的案例，介绍了置换密码和转轮密码机，目的是让读者了解代数系统在信息技术方面的应用。

1.3.5 代数系统与程序语义

作为语言，人工语言和自然语言一样，有语法、语义和语用范畴。计算机高级程序设计语言是重要的人工语言之一，它同样有语法、语义和语用范畴；进一步地，对于构成它的词汇也需要详细地定义出来，这就是所谓的词法。程序设计语言的语法是指程序的组成规则，语义是指程序的含义。语法的描述问题在形式语言理论中已经深入地进行了研究，并且得到了切实的应用。

程序设计语言的语义通常是由设计者用一种自然语言非形式地解释，实现者和使用者则依据各自的理解去实现和使用这种语言。然而，使用自然语言和非形式化的方法解释语义，容易产生歧义，造成语言设计者、用户和实现者对语义的不同理解，影响语言的正确实施和有效使用。程序设计语言中的过程调用语句就是这方面的一个典型例子：人们发现对过程调用语句的非形式解释可能导致各种不同的理解，产生多种不同的效果。

为了正确、有效地使用程序设计语言，必须了解语言中各个成分的含义，并且要求计算机系统执行这些成分所产生的效果与其含义完全一致，这种对语义精确解释的要求使得形式语义学应运而生。形式语义学的研究始于 20 世纪 60 年代初期，在程序设计语言 ALGOL 60 的设计中，第一次明确区分了语言的语法和语义，并使用巴克斯范式（Backus-Naur form，BNF）

符号系统成功地实现了语法的形式描述。语法的形式化大大推动了语义形式化的研究，围绕 ALGOL 60 的语义出现了形式语义学早期的研究热潮。以后的程序设计语言，如 Pascal、Ada 等，都有人给出了严格的形式语义，旨在为编译程序语言的编译程序提供正确依据。

语义学不像语法学那样只关心表达式的形式而不关心它们的意义，而是充分地考虑这些语言表达式在自然语言中的意义以及它们之间的关系。在语义学概念中，有真值、指派、可满足、有效模型等概念。语义学研究系统中公式的意义、系统的解释，都与模型论相关。例如，赋值语句 C=A+B 的语义是，把赋值号右边的表达式 A+B 的值赋给左边的变量 C。在编译过程中，不但要对源程序的语法检查其正确性，而且也要对语义进行检查，保证语义上的正确。在编译程序中，语义分析程序是由许多加工处理子程序组成的，其中很重要的一部分是对标识符的处理。

用程序设计语言编写的程序，规定了它对计算机系统中数据的一个加工过程，形式语义的基本方法是将程序加工数据的过程及其结果形式化，从而定义程序的语义。由于形式化中侧重面和使用的数学工具不同，形式语义主要分为四大类。

操作语义：着重模拟数据加工过程中计算机系统的操作。

指称语义：主要刻画数据加工的结果，而不是加工过程的细节。

公理语义：用公理化的方法描述程序对数据的加工。

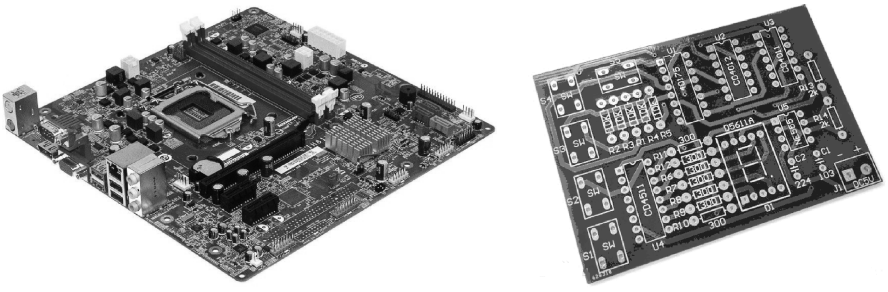
代数语义：将程序设计语言看作是刻画数据和加工数据的一种抽象数据类型，使用研究抽象数据类型的代数方法来描述程序设计语言的形式语义。

代数语义用代数方法对形式语言系统进行语义解释，即用代数公理刻画语言成分的语义，且只研究抽象数据类型的代数规格说明。抽象数据类型的代数规格说明通过构造算子和一组有关运算的代数公理刻画类型操作的行为。抽象数据类型的语法称为基调，一个基调由它的类子集（基本语法元素）和运算集（语法元素间的组合关系）两部分组成。在论证这种规格说明满足协调性和完全性的基础上，通过寻找适当的模型代数，可以定义一个抽象类型的不同层次的语义，如初始语义、终止语义等。然后就可以用普通的代数方法论证规格说明的正确性和实现的正确性。通过引入一组公理为基调指定语义，不同的公理规定不同的语义，最基本的公理形式是等式。

例如，在某些计算机的程序设计语言中，布尔类型是一种基本的数据类型，布尔类型中有两个常量“T”和“F”，对于其上的操作有与运算“and”、或运算“or”和非运算“not”。从语法的角度上来看，可以用一个代数系统<{T,F},and,or,not>来描述布尔类型。同样地，还可以给出一组公式来描述对象及操作的性质，以及对象之间的关系等，这样的公式通常用等式的形式给出，称为公理。事实上，这些公理描述了程序设计语言语句的意义，所以，代数系统集成了语法的描述，也提供了描述语义的机制，为程序设计语言语义的研究提供了一种途径。具体地讲，在语法上，bool={T,F},not:bool→bool,and:bool×bool→bool,or:bool×bool→bool；在语义上，如下的等式公理描述了操作所满足的性质：对于任意 t，u∈bool，not(true)= false，not(flase)=true，t and true = t，t and false = false，t and u = u and t，t or true=true，t or false = t，t or u = u or t。第 5 章代数系统中将详细给出布尔类型的代数规格说明。

1.3.6 印刷电路板布线问题

个人计算机都有主板，如图 1-14（a）所示。主板上连接着很多元器件，没有这些元器件的板子称为印刷线路板（printed circuit board，PCB），如图 1-14（b）所示。



（a） （b）

图 1-14 计算机的主板及印刷线路板示例

在印刷电路板出现之前，电子元件之间的互连都是依靠电线直接连接而组成完整的线路。在当代，印刷电路成为电子设备不可或缺的部分，它的主要功能是提供各项零件的相互连接。随着电子设备越来越复杂，需要的元件越来越多，印刷电路板上的导线与元件也越来越密集。

在进行印刷电路板图绘制时，首先需要确定的是印刷电路板的层数。单层板无论是成本还是制作难度都远远低于多层板，因此，绝大多数低成本的电子产品内的印刷电路板都是单层设计。但是怎样判断一个给定的电路图是否能印刷在单层印刷电路板上，而使走线间不发生交叉短路，并尽可能给出布线方案，是电路板设计中必须解决的一个问题。对于比较简单的电路拓扑结构来说，能否单层布线一目了然，但对于稍复杂的原理图，很难看出它能否通过单层布线来实现其功能。

如果把元器件看作结点，把元器件之间的连线看做边，那么一个电路图就是一个图，印刷线路板的布线问题就转化为图的可平面化问题了，这样就可以利用图的可平面化算法来检测一个电路图能否在印刷线路板上完成布线。这就是图论在电路板布线中的应用案例。

1.4 本课程的特点和学习方法

离散数学是计算机科学基础理论的核心课程之一，是计算机及应用、通信等专业的一门重要的基础课。它以研究离散量的结构和相互关系为主要目标，其研究对象一般是有限个或可数个元素，充分体现了计算机科学离散性的特点。学习离散数学的目的是为学习计算机、通信等专业的后续课程做好必要的知识准备，进一步提高抽象思维和逻辑推理的能力，为计算机的应用提供必要的描述工具和理论基础。

离散数学是建立在大量定义、定理之上的逻辑推理学科，因此，对概念的理解是学习这门课程的核心。在学习这些概念的基础上，要特别注意概念之间的联系，而描述这些联系的实体则是大量的定理和性质，因此，需要真正理解离散数学中所给出的每个基本概念的真正含义。例如，命题的定义、五个基本联结词、公式的主析取范式和主合取范式、三个推理规则以及反证法；集合的五种运算的定义；关系的定义和关系的四个性质；函数（映射）和

几种特殊函数（映射）的定义；图、完全图、简单图、子图、补图的定义；图中简单路、基本路的定义以及两个图同构的定义；树与最小生成树的定义；代数系统、群、环、域和布尔代数的定义；各类语言与各种文法、各种类型的自动机的定义；递推和递归、排列、组合和组合设计等。掌握和理解这些概念对于学好离散数学是至关重要的。

在离散数学的学习过程中，一定要注重和掌握离散数学处理问题的方法，在习题解答时，找到一个合适的解题思路和方法是极为重要的。如果知道了一道题用怎样的方法去做或证明，就能很容易地做或证出来。反之，则事倍功半。在离散数学中，虽然题型种类繁多，但每类题的解法均有规律可循。所以，在听课和平时的复习中，要善于总结和归纳具有规律性的内容，并且熟练运用这些知识。同时，还要勤于思考，对于一道题，尽可能地多探讨几种解法。

离散数学的特点是知识点集中，对抽象思维能力的要求较高。由于这些定义的抽象性，初学者往往不能在脑海中直接建立起它们与现实世界中客观事物的联系。不管是哪本离散数学教材，都会在每一章中首先列出若干个定义和定理，接着就是这些定义和定理的直接应用，如果没有较好的抽象思维能力，学习离散数学确实具有一定的困难。因此，在离散数学的学习中，要注重抽象思维能力、逻辑推理能力的培养和训练，这种能力的培养对今后从事各种工作都是极其重要的。本书中，为了改进这个问题，我们给出了一些理论成果的实际应用案例。

离散数学是理论性较强的学科，学习离散数学的关键是对离散数学有关基本概念的准确掌握，对基本原理及基本运算的运用，并多做练习。

学习离散数学的目的更多的是应用，所以，多关注数学知识的应用，既能增加自己学习的兴趣，也能从中知道数学之美。本书的每一部分内容都安排了相关的应用场景，如等价关系及等价类在粗糙集理论中的应用、关系及关系运算在数据库中的应用、图的平面化算法、最优前缀编码在文件压缩中的应用、代数系统在密码学中的应用、形式语言与自动机理论在编译系统中的应用、递归理论在程序设计中的应用、递归方程式求解在算法分析中的应用和组合理论在算法分析和组合设计中的应用等，希望读者加强这些知识的学习。

1.5　本章小结

这一章主要掌握离散数学在信息技术中的一些应用场景，了解离散数学在本专业中所处的位置，以及学习离散数学的重要性，特别是计算机类的相关专业，更要学好离散数学，为后续课程的学习打好基础。另外，还需要了解把模拟量进行数字化的过程，掌握把模拟量数字化的步骤。

1.6　习题

1. 什么是模拟量？什么是数字量？
2. 简述模拟图像数字化的过程。
3. 简述模拟音频数字化的过程。
4. 简述离散数学的研究内容及其意义。