

高等院校信息技术规划教材

数据结构 (C语言版)

朱昌杰 肖建于 编著

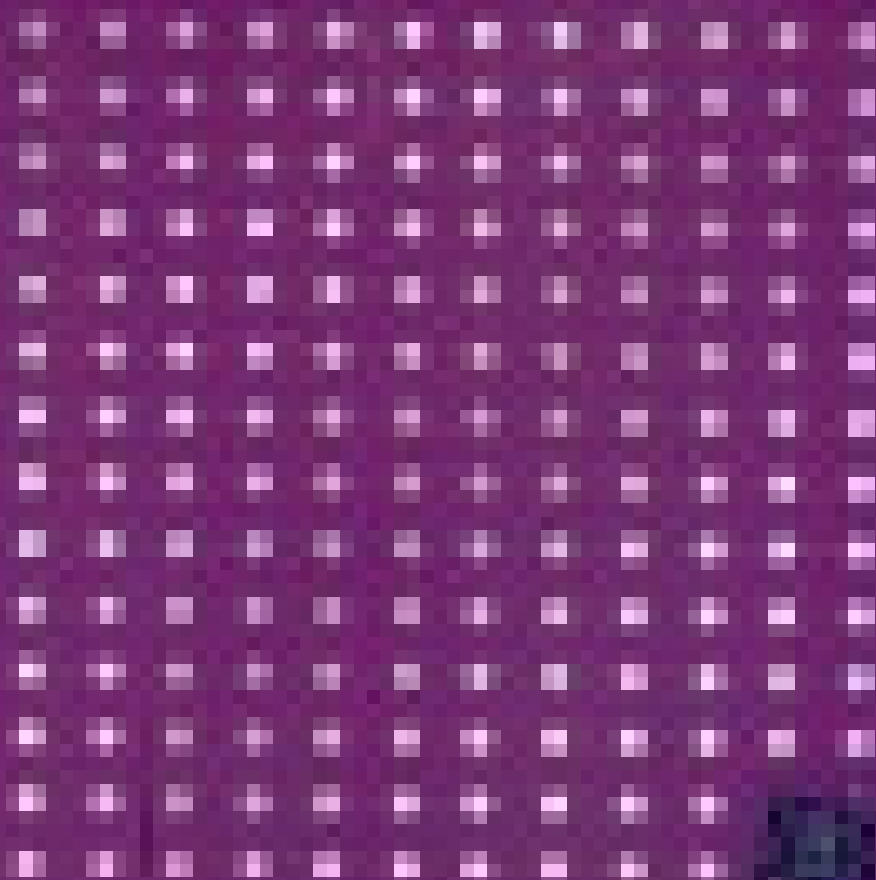


清华大学出版社

清华大学出版社 中国工信出版集团

数据结构 (C语言版)

第2版 2019年11月



清华大学出版社

高等院校信息技术规划教材

数据结构（C语言版）

朱昌杰 肖建于 编著

清华大学出版社
北京

内 容 简 介

本书系统地介绍了各种常用的数据结构与算法方面的基本知识,并使用 C 语言描述这些算法。本书共 8 章,第 1 章介绍数据结构与算法的一些基本概念;第 2~6 章分别讨论线性表、栈和队列、串、多维数组与广义表、树和二叉树、图等常用的数据结构及其应用;第 7 章和第 8 章分别介绍查找和内部排序,它们都是数据处理时广泛使用的技术。本书的特色是深入浅出,既注重理论又重视实践。全书配有大量的例题和详尽的注释,各章都有不同类型的习题。

本书可以作为高等院校计算机专业本科生的教材,也可以作为报考高等学校计算机专业硕士研究生入学考试的复习用书及专科和成人教育的教材,同时还可以作为从事计算机系统软件和应用软件设计与开发人员的参考资料。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

数据结构(C语言版)/朱昌杰,肖建于编著. —北京:清华大学出版社,2011.8

(高等院校信息技术规划教材)

ISBN 978-7-302-26043-1

I. ①数… II. ①朱… ②肖… III. ①数据结构—高等学校—教材 ②C语言—程序设计—高等学校—教材 IV. ①TP311.12 ②TP312

中国版本图书馆 CIP 数据核字(2011)第 117280 号

责任编辑:龙啟铭

责任校对:梁毅

责任印制:李红英

出版发行:清华大学出版社

地 址:北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

邮 编:100084

社 总 机:010-62770175

邮 购:010-62786544

投稿与读者服务:010-62795954,jsjje@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015,zhiliang@tup.tsinghua.edu.cn

印 刷 者:北京富博印刷有限公司

装 订 者:北京市密云县京文制本装订厂

经 销:全国新华书店

开 本:185×260

印 张:14.5

字 数:344 千字

版 次:2011 年 8 月第 1 版

印 次:2011 年 8 月第 1 次印刷

印 数:1~3000

定 价:25.00 元

产品编号:041764-01

前 言

数据结构是计算机科学中的一门综合性的专业基础课，是介于数学、计算机硬件和计算机软件三者之间的一门核心课程。这门课程的内容不仅是一般程序设计（特别是非数值性程序设计）的基础，而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序的重要基础。学好该课程，不仅对这些后续课程的学习有很大帮助，而且在实际中也有广泛的用途。

本书是编者针对数据结构课程概念多、算法灵活和抽象性强等特点，在总结长期教学经验的基础上编写的。本书共 8 章，第 1 章介绍数据结构和算法的基本概念及本书用到的 C 语言的知识；第 2 章为线性表，介绍线性表的两种存储结构及基本操作的实现；第 3 章为栈和队列，介绍这两种特殊的线性结构的概念与应用；第 4 章为串、多维数组与广义表，介绍串的概念与模式匹配算法，多维数组和稀疏矩阵的概念与相关运算的实现，广义表的定义与基本操作；第 5 章为树形结构，介绍树和二叉树的概念与各种运算的实现过程，其中特别突出二叉树的各种算法实现；第 6 章为图结构，介绍图的概念和图的各种算法实现及应用；第 7 章和第 8 章分别介绍查找和内部排序常用的各种方法。为了便于学习和巩固知识，在每章后配有大量习题供读者练习。

本书是计算机科学与技术专业国家级和省级特色专业建设点、“计算机基础系列课程教学团队”省级教学团队、《数据结构》精品课程建设成果之一；也是数据结构课程组教师多年来在数据结构课程教学研究和教学改革中的经验与成果的结晶。本书由朱昌杰、肖建于主要编写，张震、李璟、刘怀愚、陈美荣、沈龙凤、赵娟、施汉琴等老师参与了部分章节的编写。研究生吴波、任逸卿同学对程序的调试和校对工作付出了大量的劳动。全书由朱昌杰、肖建于负责统稿。本书出版过程中得到了清华大学出版社的大力支持，在此表示衷心的感谢！

尽管编者在写作过程中非常认真和努力，但由于编者学识有限，书中难免存在错误和不足之处，恳请各位专家和广大读者批评指正，不胜感激。

编 者
2011 年 5 月

目 录

第 1 章 绪论.....	1
1.1 数据结构的基本概念.....	1
1.1.1 数据的逻辑结构.....	1
1.1.2 数据的物理结构.....	2
1.2 数据类型和抽象数据类型.....	3
1.2.1 数据类型.....	3
1.2.2 抽象数据类型.....	4
1.3 C 语言的数据类型.....	5
1.4 用 C 语言描述算法的注意事项.....	6
1.4.1 算法及其特征.....	6
1.4.2 C 语言描述算法的注意事项.....	7
1.5 算法设计目标和算法效率度量.....	9
1.5.1 算法设计目标.....	9
1.5.2 算法效率度量.....	9
习题 1.....	11
第 2 章 线性表.....	13
2.1 线性表的类型定义.....	13
2.1.1 线性表的定义.....	13
2.1.2 线性表的基本运算.....	14
2.2 线性表的顺序存储及实现.....	14
2.2.1 顺序表.....	14
2.2.2 顺序表基本运算的实现.....	16
2.2.3 顺序表其他算法举例.....	19
2.3 线性表的链式存储及实现.....	20
2.3.1 单链表.....	20
2.3.2 单链表的基本运算的实现.....	22
2.3.3 单链表的其他操作举例.....	28
2.3.4 循环单链表.....	30
2.3.5 双向链表.....	31
2.3.6 静态链表.....	34
2.4 线性表的应用举例.....	38
2.4.1 一元多项式的表示与相加.....	38

2.4.2 约瑟夫环问题.....	41
习题 2.....	42
第 3 章 栈和队列.....	45
3.1 栈.....	45
3.1.1 栈的定义.....	45
3.1.2 栈的顺序存储结构.....	46
3.1.3 栈的链式存储结构.....	48
3.2 栈的应用举例.....	49
3.2.1 数制转换.....	50
3.2.2 括号匹配的检验.....	50
3.2.3 简单表达式求值.....	52
3.3 队列.....	56
3.3.1 队列的定义.....	56
3.3.2 队列的顺序存储结构.....	57
3.3.3 队列的链式存储结构.....	60
3.4 队列应用举例.....	63
3.4.1 键盘输入循环缓冲区问题.....	63
3.4.2 舞伴问题.....	64
习题 3.....	66
第 4 章 串、多维数组与广义表.....	69
4.1 串.....	69
4.1.1 串的定义.....	69
4.1.2 串的实现.....	71
4.1.3 串的模式匹配算法.....	75
4.1.4 串的操作应用——文本编辑.....	76
4.2 数组.....	78
4.2.1 数组的定义.....	78
4.2.2 数组的顺序存储结构.....	79
4.3 矩阵的压缩存储.....	80
4.3.1 特殊矩阵.....	80
4.3.2 稀疏矩阵.....	82
4.4 广义表.....	86
4.4.1 广义表的定义.....	86
4.4.2 广义表的基本操作.....	87
4.4.3 广义表的存储结构.....	87
习题 4.....	89
第 5 章 树和二叉树.....	91
5.1 树的基本概念.....	91

5.1.1	树的定义.....	91
5.1.2	树的基本术语.....	93
5.2	二叉树.....	94
5.2.1	二叉树的定义.....	94
5.2.2	二叉树的性质.....	96
5.2.3	二叉树的存储结构.....	98
5.3	遍历二叉树和线索二叉树.....	100
5.3.1	二叉树的遍历方法及递归实现.....	100
5.3.2	二叉树遍历的非递归实现.....	102
5.3.3	遍历算法的应用.....	105
5.3.4	由遍历序列构造二叉树.....	108
5.3.5	线索二叉树.....	109
5.4	树和森林.....	112
5.4.1	树的存储结构.....	112
5.4.2	树与二叉树的转换.....	114
5.4.3	森林与二叉树的转换.....	115
5.4.4	树和森林的遍历.....	117
5.5	哈夫曼树.....	118
5.5.1	哈夫曼树的基本概念.....	118
5.5.2	哈夫曼树的构造方法.....	119
5.5.3	哈夫曼编码.....	122
习题 5	124
第 6 章	图.....	126
6.1	图的基本概念.....	126
6.1.1	图的定义.....	126
6.1.2	图的基本术语.....	127
6.2	图的存储结构.....	128
6.2.1	邻接矩阵.....	128
6.2.2	邻接表.....	129
6.3	图的遍历.....	130
6.3.1	深度优先搜索.....	130
6.3.2	广度优先搜索.....	132
6.4	最小生成树.....	134
6.4.1	最小生成树的基本概念.....	134
6.4.2	Prim 算法构造最小生成树.....	135
6.4.3	Kruskal 算法构造最小生成树.....	137
6.5	有向无环图及其应用.....	139
6.5.1	拓扑排序.....	139

6.5.2	关键路径.....	141
6.6	最短路径.....	143
6.6.1	单源点最短路径.....	143
6.6.2	每对顶点之间的最短路径.....	145
习题 6	147
第 7 章	查找	151
7.1	查找的基本概念.....	151
7.2	顺序表的静态查找.....	152
7.2.1	顺序查找.....	152
7.2.2	折半查找.....	153
7.2.3	分块查找.....	156
7.3	树表的动态查找.....	158
7.3.1	二叉排序树.....	159
7.3.2	平衡二叉树.....	165
7.3.3	B-树.....	171
7.3.4	B ⁺ 树.....	176
7.4	散列 (Hash) 表的查找.....	178
7.4.1	散列表的基本概念.....	179
7.4.2	散列函数构造方法.....	179
7.4.3	处理冲突的方法.....	182
7.4.4	散列表的查找及其分析.....	184
习题 7	187
第 8 章	内部排序	190
8.1	排序的基本概念.....	190
8.2	插入排序.....	191
8.2.1	直接插入排序.....	191
8.2.2	折半插入排序.....	193
8.2.3	希尔排序.....	194
8.3	交换排序.....	196
8.3.1	冒泡排序.....	196
8.3.2	快速排序.....	198
8.4	选择排序.....	202
8.4.1	直接选择排序.....	202
8.4.2	堆排序.....	204
8.5	归并排序.....	207
8.6	基数排序.....	210
8.6.1	多关键码排序.....	211

8.6.2 链式基数排序.....	212
8.7 各种内部排序算法的比较.....	216
8.7.1 内部排序算法的性能比较.....	216
8.7.2 各种内部排序算法的选择.....	218
习题 8.....	219
参考文献.....	222

第 1 章 绪 论

本章知识要点:

- 数据结构的基本概念
- 逻辑结构、物理结构的分类及其特点
- 数据类型和抽象数据类型的概念及其定义方式
- C 语言的常用数据类型
- 用 C 语言描述算法的注意事项
- 算法设计的目标和算法效率的度量

1.1 数据结构的基本概念

随着计算机技术的飞速发展, 计算机的应用已经遍布人类社会的各个领域。计算机处理的数据也越来越丰富, 而数据的组织形式和表示方式直接关系到计算机对数据的处理效率, 因此, 为了更好地进行程序设计, 需要对程序所加工处理的对象进行系统和深入的研究。而数据结构就是要研究各种数据的特性以及数据之间的关系, 进而根据实际情况合理地组织、存储数据, 最终设计出好的算法。

数据是对客观事物的符号表示, 在计算机科学中是指所有能输入到计算机中并被计算机程序所处理的符号的总称, 它是计算机程序加工的“原料”。例如, 一个班级的所有学生的成绩、某学院所有教师信息记录、从公元 100—3000 年间的所有闰年以及某公司所有职员的薪资信息等都称为数据。

数据元素是数据的基本单位, 在程序设计中通常作为一个整体进行考虑和处理。例如, 学生学籍管理系统中关于某个学生的一条记录就是一个数据元素, 每个记录中可能包含多个属性(如姓名、学号、专业、年级等), 每个属性就称为一个数据项, 数据项是具有独立含义的数据的最小单位。

数据结构是相互之间存在一种或多种特定关系的数据元素的集合。这些数据元素不是孤立存在的, 而是有着某种关系的, 这种关系称为结构。数据结构包括的内容有数据元素之间的逻辑结构、数据在计算机中的存储方式(即物理结构)和施加在该数据上的操作。

1.1.1 数据的逻辑结构

数据元素与数据元素之间的逻辑关系称为数据的逻辑结构。根据数据元素之间逻辑关系的不同特性, 可将数据结构分为以下 4 类基本结构。

1. 集合

这种结构中的数据元素同属于一个集合，除此之外别无其他关系（集合在数据结构中很少讨论）。

2. 线性结构

这种结构中的数据元素之间存在一个对一个的关系。

3. 树形结构

这种结构中的数据元素之间存在一个对多个的关系。

4. 图形结构

这种结构中的数据元素之间存在多个对多个的关系。

从上述关系来看，线性结构是树形结构的特例，树形结构是图形结构的特例。一般情况下，数据结构的逻辑结构可以用二元组来表示：

$$\text{Data_Structure}=(D,R)$$

其中 D 是数据结点的有限集合， R 是 D 上的关系的有限集合，其中每个关系都是从 R 到 D 的关系。

下面将给出关于线性结构、树形结构和图形结构的 3 个实际用例，以帮助读者深入理解逻辑结构。

例 1-1 某班级学生学籍管理问题。当我们要查阅张伟同学的家庭住址时，或者当我们要查询张伟同学是哪个民族时，因为一个班级里可能有两位同学都叫张伟，所以我们要按学号进行查询，而同名的两个张伟的学号是不同的。所有同学的学号存在一个对一个的关系，这是一种典型的线性关系，如图 1.1 所示。

20101204038	王平	男	汉族	天津市河西区幸福大街 36 号
20101204039	闻亮亮	女	满族	淮南市八公山区西四村 64 栋 23 号
20101204040	张伟	男	布依族	唐山市古冶区平庄村 128 号
20101204041	张伟	男	汉族	淮北市濉溪县王桥 100 号
20101204042	仲萍	女	汉族	邯郸市邯山区滏河北大街 56 号

图 1.1 学生学籍线性结构图

例 1-2 计算机中的文件和文件夹的组织问题。计算机中某个磁盘可能包含多个文件夹，每个文件夹又可以包含多个子文件夹，每个子文件夹又可以包含自己的文件和子文件夹，文件夹之间或文件夹与文件之间存在着一个对多个的关系，这是一种典型的树形结构，如图 1.2 所示。

例 1-3 连锁超市的分店问题。某家大型连锁超市在某城市开了 6 家分店，各个分店除了和总店有一定关系外，还和其他分店存在着某种联系，所以各个连锁店之间存在一种多个对多个的关系，这是一种典型的图形结构，如图 1.3 所示。

1.1.2 数据的物理结构

数据的物理结构即是数据在计算机中的存储方式，数据的存储方式将直接影响或决定数据的处理效率。常用的存储结构有以下 4 种。

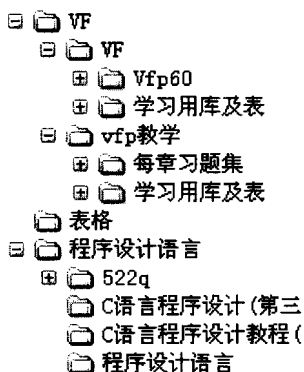


图 1.2 文件夹树形结构图

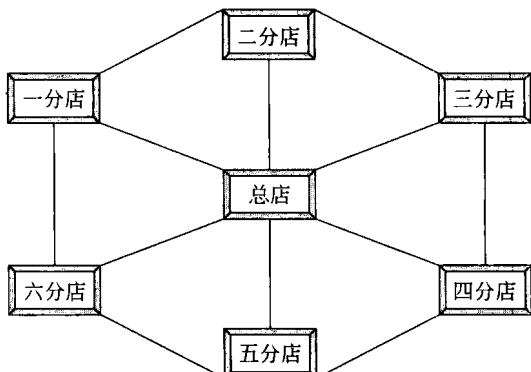


图 1.3 连锁超市图形结构图

1. 顺序存储结构

这种存储方式是将逻辑上连续的数据存储在地址连续的内存空间中，存储地址的相对关系即对应数据之间的逻辑关系。其优点是节省存储空间，因为不需要额外的存储空间来保存数据间的逻辑关系，可以进行随机访问。其缺点是进行数据的插入和删除时可能需要移动大量的结点。

2. 链式存储结构

这种存储方式中数据的存储结构不表示数据之间的逻辑结构，结点之间的逻辑结构是由附加的指针字段来表示的。其优点是在进行数据的插入和删除时，仅需修改指针，不需要移动结点。其缺点是需要额外的存储空间来存储结点之间的逻辑关系。

3. 索引存储结构

这种存储方式在存储结点信息的同时，还建立附加的索引表。索引表由索引项构成，其一般形式为：(关键字，地址)，关键字能够唯一地标识一个结点，地址表示指向结点的指针。其优点是在进行插入和删除操作时，只需移动存储在索引表中对应结点的存储地址，而不必移动存放在结点表中结点的数据，因此保持了较高的数据修改运算效率。其缺点是创建和维护索引表要增加时间和空间的开销。

4. 散列存储结构

这种存储方式是根据结点的关键字通过哈希函数计算出的值来确定结点的存储地址。其优点是查找速度快，只要给出待查找的关键字，即可计算出该结点的存储地址。与上述三种方法不同的是，散列存储方式只适合要求对数据进行快速查找和输出的场合，其关键是要选择一个好的散列函数和处理“冲突”的方法，本书的第7章将详细介绍这种方法。

1.2 数据类型和抽象数据类型

1.2.1 数据类型

在用高级程序设计语言编程序时，要对程序中用到的每一个常量、变量和表达式定

义数据类型, 不同类型数据的取值范围不同, 所允许进行的操作也不同。因此数据类型是一个值的集合和定义在该值集上的一组操作的总称。例如在 C 语言中, 基本整型一般为 2 个字节 (即 16 位), 表示范围为 $-2^{15} \sim (2^{15}-1)$, 即 $-32\,768 \sim 32\,767$, 允许进行的操作有加、减、乘、除和取模等, 并且定义了运算规则。

在高级程序设计语言中, 数据类型分为两类: 原子类型和结构类型。原子类型又称为非结构类型, 其值是不可分解的, C 语言中的原子类型即是其基本类型 (整型、字符型、浮点型和枚举类型)、指针类型和空类型。结构类型又称为构造类型, 该类型的各个成分既可以是原子类型的, 也可以是结构类型的 (即允许嵌套定义)。C 语言中的结构类型是数组类型、结构体类型和共用体类型等。

1.2.2 抽象数据类型

抽象数据类型 (Abstract Data Type, 简称 ADT) 是指用于表示实际应用问题的数据模型以及定义在该模型上的一组操作。抽象数据类型与一般数据类型的概念是一致的, 都只专注于其逻辑特性, 相同的数据类型在不同的计算机系统上的表示和实现可能是不同的, 但只要其数学特性不变, 就不会影响外部的使用。

另一方面, 从抽象的层次和定义的范畴来看, 抽象数据类型较之一般数据类型抽象层次更高, 定义范畴更广。抽象数据类型不再局限于计算机系统中已经实现的数据类型, 可以是用户所定义的数据类型, 它可以由计算机系统已经实现的数据类型来表示和实现。抽象数据类型必须先定义后使用, 定义抽象数据类型只描述数据的逻辑结构及允许进行的操作, 不考虑数据的物理存储及其操作的具体实现。

一个具体问题的抽象数据类型定义一般包括数据对象 (即数据元素的集合)、数据关系和基本操作三方面的内容。抽象数据类型可用三元组 (D, R, P) 来表示。其中 D 是数据对象; R 是 D 上的关系的集合; P 是 D 中数据运算的基本操作集合。对抽象数据类型的定义需要约定一定的格式, 本书均采用如下格式定义抽象数据类型:

```
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
```

其中基本操作的定义格式为:

```
基本操作名 (参数列表)
    初始条件
    操作结果
```

基本操作有两种参数: 赋值参数只为操作提供输入值; 引用参数以 & 打头, 除了可以提供输入值外, 还将返回操作结果。“初始条件”是指操作执行之前数据结构和参数应该满足什么条件, 不满足条件则操作失败, 程序将返回出错信息。“操作结果”是指操作

正常完成的情况下数据结构的变化及应返回的结果。

例 1-4 抽象数据类型复数的定义。

```
ADT Complex
{
    数据对象 D: D={c1, c2 | c1, c2 均为实数}
    数据关系 R: R={<c1, c2> | c1 是复数的实数部分, c2 是复数的虚数部分}
    基本操作 P:
        InitComplex (&Z, v1, v2)
            操作结果: 构造了复数 Z, 元素 c1 和 c2 分别被赋值为参数 v1 和 v2
        DestroyComplex (&Z)
            操作结果: 复数 Z 被销毁
        GetReal (Z, &real)
            操作结果: 用 real 返回复数 Z 的实部的值
        GetVirtual (Z, &virtual)
            操作结果: 用 virtual 返回复数 Z 的虚部的值
        Add (Z1, Z2, &sum)
            操作结果: 用 sum 返回两个复数 Z1, Z2 的和
        ModComplex (Z, &model)
            操作结果: 用 model 返回复数 Z 的模
} ADT Complex
```

1.3 C 语言的数据类型

C 语言提供了如图 1.4 所示的一些数据类型, 并且可以由这些数据类型构造出不同的、更加复杂的数据类型。

下面将对各种类型的变量作简单介绍。

1. 整型

整型一般用 `int` 表示, 其变量可以分为基本整型 (`int`)、短整型 (`short int` 或 `short`) 和长整型 (`long int` 或 `long`)。一个基本整型变量值的表示范围为 $-32\,768 \sim 32\,767$ 。若变量的值不为负值, 还可以定义其为无符号整型; 若未加声明, 则表示为有符号整型。C 语言没有具体规定各类整型数据所占用的内存字节数, 只要求长整型数据长度不短于基本整型, 短整型不长于基本整型。具体如何实现因计算机的不同可能有所不同。

2. 浮点型

浮点数即实数, 浮点型一般用 `float` 表示, 浮点型变量可以分为单精度 (一般占 4 字节, 用 `float` 表示)、双精度 (一般占 8 字节, 用 `double` 表示) 和长双精度 (一般占 16

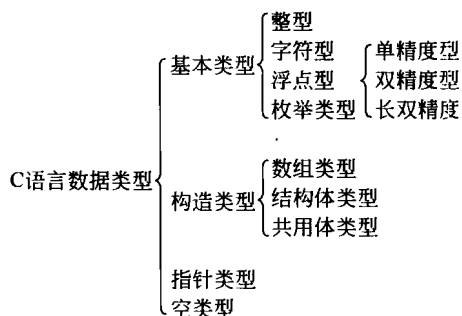


图 1.4 C 语言的数据类型

字节, 用 `long double` 表示)。不同类型的浮点数的有效数字长度不同, 所表示的数值范围也不同, 具体请读者参阅相关 C 语言教程, 本书不做详细说明。

3. 字符型

C 语言的字符常量是用单撇号括起来的一个字符, 一般用 `char` 来表示。在内存中存放字符并不是存放字符本身, 而是将该字符的 ASCII 码放到存储单元中。字符串常量是一对双撇号括起来的字符序列。C 语言规定 `'\0'` 作为字符串的结束标志, 该标志由系统自动添加, 在书写时不需要添加, 输出时也不会输出该字符。

4. 枚举类型

枚举类型一般适用于变量的值只有几种可能的情况, 可以将变量的值一一列举出来, 变量只能取该范围内的其中一个值。枚举类型的声明一般用 `enum`。

5. 数组类型

数组是由用户定义的结构数据类型, 是按一定顺序排列的具有相同性质的变量的集合。所有数组元素具有相同的数组名, 但有不同的数组下标, 下标用方括号括起来。数组元素可以是基本类型, 也可以是构造类型的。

6. 结构体类型

结构体类型是将不同类型的数据组合在一起构成新的类型, C 语言允许用户根据在编程时的需要自行定义该类型。结构体类型一般以 `struct` 打头, 必须先声明结构体类型再定义该类型的变量。

7. 共用体类型

共用体类型一般用 `union` (有些 C 语言书籍将其直译为联合) 来表示, 共用体类型变量的内存长度等于最长的成员的长度。在程序中不能引用共用体变量, 只能引用共用体变量中的成员, 在每一瞬间只能有一个成员起作用, 即各成员不是同时存在和起作用的。

8. 指针类型

定义指针变量需要用 `*` 符号, 指针变量中存放的不是数据本身, 而是反映数据存放位置的地址。指针可以应用到链表中、数组中、结构体类型数据的成员中等, 这是编程者需要重点掌握的一个知识点。

9. 空类型

空类型一般用 `void` 来表示。在调用函数值时, 通常应向调用者返回一个函数值。但是有一类函数调用后不需要向调用者返回函数值, 这种函数可以定义为空类型。

1.4 用 C 语言描述算法的注意事项

1.4.1 算法及其特征

算法是为了解决特定问题而规定的指令的有限序列, 每一条指令表示一个或多个操作。算法具有以下 5 个重要特性。

1. 有穷性

一个算法必须总是（对任何合法的输入值）在执行有穷步之后结束，且每一步都可以在有穷时间内完成。有穷时间是指可以接受的时间内，例如一个程序能够执行并完成，但是需要执行 100 年，则我们认为其是无法实现的。

2. 确定性

对每种可能的情况所执行的操作，算法中都必须有确切的规定，使程序的阅读者和执行者能够明确理解其含义，并且在任何条件下，算法都要保证相同的输入只能得到相同的输出。例如这样的描述“当 A 属于中年时，salary 为 2000 元”，假如 38 岁算是中年，41 岁也算是中年，那么 46 岁算不算呢？该条件无法判断是否满足，因此，如果算法中出现了这样的判断条件，则程序将不能正确执行。

3. 可行性

算法中的每一个操作都必须是能够准确执行的。例如，如果出现 $B=3$ ，则 $A/(B-3)$ 就是无法执行的。

4. 输入

一个算法有 0 个或多个输入。如果需要初始值，可以在程序运行时由用户输入，也可以在算法中直接给出。

5. 输出

一个算法有 1 个或多个输出。

在算法的上述 5 个重要特性中，最基本的是有穷性、确定性和可行性。

1.4.2 C 语言描述算法的注意事项

1. 预定义常量

```
#define TRUE      1
#define FALSE    0
#define MAXSIZE  1000
#define OK       1
#define ERROR    0
#define INFEASIBLE -1
#define OVERFLOW -2
```

2. 输入和输出语句

```
scanf(格式控制字符串, 输入项表);
printf(格式控制字符串, 输出项表);
```

3. 赋值语句

```
变量名=表达式;
```

4. 选择语句

```
if(条件)
```