数据结构与算法—C语言程序设计

姜文清 编著
徐良贤 主审

数据结构与算法
—C语言程序设计

上海交通大学出版社

数 据 结 构 与 算 法

——C 语言程序设计

姜文清　编著

徐良贤　主审

上 海 交 通 大 学 出 版 社

内 容 提 要

本书是数据结构与算法的教材，书中出现的程序都是用C语言编写的。全书共分六章。第一章简明地介绍C语言；第二章介绍各种线性表的结构、有关的操作算法和应用实例；第三章介绍文本编辑；第四章介绍树和二元树；第五章介绍图的表示方法、搜索算法以及求解路径和最小连接树问题；第六章介绍八种分类算法和一个通用分类程序。

本书是为计算机类专业大学生、研究生编写的教材和教学参考书，也可作为工程技术人员的"数据结构与算法"程序手册。

数据结构与算法

上海交通大学出版社出版
（淮海中路1984弄19号）
新华书店上海发行所发行
江苏常熟文化印刷厂印装

开本 787×1092 毫米 1/16 印张 13.25 字数 320,000
1988年8月第1版 1988年8月第1次印刷
印数 1—6,000

ISBN 7-313-00101-0/TP·3 科技书目：167—286

定价：2.20元

前　言

数据结构是计算机专业的基础课程之一。研究数据结构与算法及其实现是计算机科学的核心问题。本课程的设置，旨在使学生建立起“结构”与“算法”的模型概念，培养学生设计算法，开发程序的实际能力，为从事计算机科学的研究和工程设计奠定牢固的基础。

本书在讲解数据结构时，结合C语言程序设计，把数据结构及其有关的操作和算法，系统地贯穿在完整的实用程序之中，既介绍基本原理，又提供实现范例。

C程序设计语言，由于其数据类型丰富，语句精炼、灵活，效率高，表达力强，以及可移植性好等许多优点，受到了世界上软件专家们的赞赏。在我国，许多数理工科高等院校的有关专业都开设了C语言课程。本书用C语言编写数据结构与算法的描述程序，不仅使算法清晰，而且还能给学生提供数据结构在计算机存储器中的映象。

本书第一章简明地介绍C语言，此外，还介绍一种重要的程序设计方法——递归。第二章介绍各种线性表的结构、有关的操作算法和应用实例。第三章介绍文本编辑，包括读、写文件的方法和两个实用程序，介绍这两个程序所采用的数据结构和主要编辑操作的算法。第四章介绍关于树和二元树。第五章介绍图的表示方法，搜索算法以及求解路径和最小连接树问题。第六章介绍八种分类算法和一个通用分类程序。本书各章的题材广泛、内容丰富，对每种数据结构和算法都有相应的实现程序，对算法较为复杂的程序，除配有图示说明之外，还通过典型数据给出执行程序的屏幕记录。本书中的程序，体现了结构化程序设计的思想，有较好的程序风格。在使用本书时，教师可根据教学学时数选择其中主要的内容，其余部分留给学生自己阅读。

本书是为计算机类专业大学生、研究生编写的教材或教学参考书。也可作为工程技术人员的“数据结构与算法”程序手册。

在编写本书的过程中，得到了哈尔滨工业大学郭福顺教授热情的关切与鼓励。上海交通大学徐良贤副教授详细地审阅了本书初稿的全文，提出了许多修改意见。对此，编者表示衷心的感谢。

由于本人学识水平有限，书中不妥之处在所难免，敬请各位专家、读者批评指正。

编　者

1987年3月

目　录

第一章 引 论

1.1 数据和数据结构

数据是计算机程序使用和处理的原始资料或信息，是数字、符号或事件的集合；数据结构指的是数据的组织形式。比如，向量和矩阵是不同的数据结构；一个身份证上所列的各项：姓名、性别、出生年月日、工作单位、职务以及住址等信息，构成结构型的数据；各级行政机关(如省、地、市、县、乡等)构成树形的数据结构；一张交通图或程序流程图构成图结构。

我们把客观事物本身的结构关系称为逻辑结构。这里，我们对客观事物本身的属性不感兴趣，只关心它的结构形式。我们把抛开客观事物本身只在形式上相同的这类数据结构称为抽象结构。比如，线性表、树、二元树和图等。抽象结构在计算机中的映象即为存储结构。对同一种抽象结构，不同的实现可以采用不同的存储结构。

1.2 C程序设计语言概述

1.2.1 简单程序举例

为了尽快地了解C语言，我们先看一个简单的程序。下列程序为读输入一行字符，然后按相反顺序输出：

```
main( )
{
    char s[80],  c;
    int t;
    t = 0;
    while((c = getchar( )) != '\n')
        s[++t] =c;
    while(t != 0)
        putchar(s[t--]);
}
```

C程序由若干段组成，每个段称为函数。本程序只有一个函数main，main是C规定的函数名，用来标识一个主程序段，是开始执行程序的入口。除此之外，函数名都由程序员自行指定的标识符组成。程序中的一对花括号{ }标识函数体的开始和结束。C的语句是用分号分隔的。若干语句可以构成一个复合语句，用一对花括号括起来。程序中的char和int是类型关键字，说明其后的数组或变量的类型分别为字符型和整型；s是数组名，数组的长度为80，数组下标从0开始；c是字符变量，其值为一个字符；t是一个整形变量。语句t=0表示t的初值为0；while是循环语句的保留字，其后圆括号中的内容为循环条件；c=getchar( )是赋值表达式，getchar是标准输入函数，每次读输入一个字符，并以该字符为返回值；'\n'如为换行字符；!=是不等号。执行第一个循环的条件为输入字符不是换行符；执行第二个循环的条件为t的值不等于0。语句

```
s[++t]=c;
```

表示先将 t 的值加 1，然后将 c 的值赋给以 t 的当前值为下标的数组元素。putchar 是标准输出函数，该函数将参数字符写入输出。语句

```
putchar(s[t--]);
```

表示将数组元素 s[t]的值写入标准输出，然后下标 t 的值减 1。

1.2.2 基本数据类型

1.2.2.1 常数

C 的常数有四种基本类型：

整常数 由数字串组成。若以数字 0 开头，则看成是八进制数。其中的 8 和 9 看成是八进制的 10 和 11；若以 0x 或 0X 开头，则看成是十六进制数。其余情况为十进制数。若其值超过机器所能表示的带符号数的最大范围，则看成是长整型数。

浮点常数 浮点常数由整数部分、小数部分和小数点组成，也可以由 e 或 E 以及相应的指数部分，构成"科学记数法"的浮点数，如 1.25e－2 表示 1.25×10^{-2}。每个浮点常数都是双精度数。

字符常数 字符常数是单引号 '' 内的单个字符，如 'x'，其值是该字符在机器字符集里的编码。有些字符前置反斜线转义之后具有特殊意义：

'\0' 空字符(不是空白符)
'\t' 水平制表
'\n' 换行
'\"' 双引号
'\\' 反斜线

字符串 字符串是双引号" "内的字符序列。例如"The result is: %d\n"，双引号不属于字符串范围。编译程序在每个字符串的末尾填一个空字符'\0'，以示结束。

1.2.2.2 标识符、变量

C 的标识符是字母开头的字母数字串，下划线'_'看成是字母。在 C 语言中，有以下七种基本类型的说明保留字

char	字符型
int	整　型
short	短整型
long	长整型
unsigned	无符号整型
float	浮点型
double	双精度型

例如：

```
char c, s[5];float x, y, z;
```

在类型说明中，int 前面还可以冠有 short、long 或 unsigned，其结果和没有 int 时相同。

还可以用类型定义保留字 typedef 定义某个标识符为类型名，比如，用了

```
typedef double D;
```

后，即可把D作为说明双精度变量的类型名。

例如：

```
D x,y,z;
```

其作用和

```
double x,y,z;
```

相同。

在C程序中，char类型的数据可以用int类型的变量表示。如果在同一个算术表达式中存在上述不同类型的数据，则在运算时，C语言自动进行类型转换，把较短类型的数据转换成较长类型，运算结果也是较长类型。字符类型的数据转换成数值时，其结果是该字符的代码值。在一个变量的前边还可以加注类型名，表示强制其类型转换。比如，在变量说明中有

```
double x;
```

而在运算过程中，用(int)x的形式，其结果是一个整数值，但x本身的值没有变。

1.2.3 运算符

1.2.3.1 算术运算符

二元算术运算符有：+、-、*、/和%。其中+、-、*和/表示加、减、乘和除，%是取余数运算符，即两个整数相除，结果为除得的余数。例如10%3，其结果是1。当a、b均为整数时，且b≠0，则(a/b)*b+a%b恒等于a。

此外，还有一元运算符'-'，即取负。

在上述运算中，负号的优先级最高，其余按数学规则排列其优先顺序。

1.2.3.2 关系运算符

对应于数学中的关系符号，C语言的关系运算符如下：

C的关系符	数学关系符
>	>
>=	≥
<	<
<=	≤
==	=
!=	≠

其中，运算符==和!=的优先级低于前四个运算符。关系运算的结果，若为真，产生整数1；否则，产生整数0。

1.2.3.3 逻辑运算符

逻辑运算只检查操作数是否为0。若不为0，则为逻辑真；否则为逻辑假。若运算的结果为真，则其值为整数1，否则为整数0。逻辑运算符如下：

&&	逻辑与
||	逻辑或
!	逻辑非

其优先级按!、&&、||的顺序递减。

1.2.3.4 位式运算符

位式运算符检查操作数的每一个二进制位，按每一位的值进行运算。有如下运算符：

&	位式与
|	位式或
^	异或，即按位加
~	取反码
<<	左移位，如x<<n，是将x的每一位向左移n位(整数)，空出的位填零
>>	右移位，如x>>n，是将x的每一位向右移n位。若x是无符号数，则空出的位填零(逻辑移位)，否则空出的位填符号位的值(算术移位)

移位运算中的两个操作数必须都是整数。

位式运算符的优先级按~、移位、&、^、|的顺序递减。

1.2.3.5 特殊的一元运算符

下列一元运算符用于实现某种变换。其中，V表示可保存值的量，E表示表达式，T表示类型名。

*E	指针E所指存储单元的内容
&V	V所在存储单元的地址
sizeof E	E占用存储单元的字节数
sizeof(T)	T类型量占用存储单元的字节数
++V	先将V的值增1，然后存取V
V++	先存取V的值，然后将V的值增1
--V	先将V的值减1，然后存取V
V--	先存取V的值，然后将V的值减1

全部一元运算符都按从右至左的顺序结合。就是说，当这些运算符连续出现在一个表达式中时，右边算符的优先级高于左边算符。

1.2.3.6 条件运算符

一个条件表达式有两个算符，三个操作数。其形式如下：

E1 ? E2 : E3

其中，E1、E2和E3都是表达式。若E1的值不为0，则该条件表达式的结果是E2的值，否则为E3的值。

1.2.3.7 赋值运算符

诸如FORTRAN语言中的赋值语句，在C语言中称做赋值表达式。在赋值表达式中，左操作数是一个能保存值的量，右操作数是一个表达式。复合赋值运算，即多重赋值，要按从右向左的顺序结合。我们用V表示一个能保存值的量，用E表示一个表达式，赋值运算符及其构成的表达式有如下的形式：

V = E

其意义是把E的结果值赋给V；

V += E

如果E是一个算术表达式，则使V的值增加E，结果赋给V，这种形式和

V = V + E

是等价的。类似地，上述二元算术运算符－、*、/和%以及二元位式运算符＞＞、＜＜、&、|和∧都可以在其后跟一个等号构成赋值运算符。例如：

```
i %= 3
```

其意义是将i的值除以3（整数相除），然后将余数赋给i；

```
i >>= 3
```

是将i右移3位之后的结果赋给i，它和

```
i = i >> 3
```

是等价的。

1.2.3.8 运算符的优先顺序

表1.1列出了C语言的全部运算符及其优先顺序。其中，同一行的运算符具有相同的优先级。在纵向上，第一行运算符的优先级最高，以下逐行递减。表中的"结合性"指的是同一行运算符连续出现时的优先顺序。

表1.1 运算符的优先顺序

运　　算　　符	结合性
( ) [ ] -> （见1.3节）	从左至右
! ～ ++ -- －（负） (type) *(一元) &(一元) sizeof	从右至左
* / %	从左至右
+ －	从左至右
<< >>	从左至右
< <= > >=	从左至右
== !=	从左至右
&	从左至右
∧	从左至右
|	从左至右
&&	从左至右
||	从左至右
?:	从右至左
= += -= *= /= %= >>= <<= &= |= ∧=	从右至左
, （见1.2.6.2节）	从左至右

1.2.4 数据输入和输出

在C语言程序中，数据的输入和输出是通过调用标准I/O函数实现的。我们把从终端键盘上输入数据称为标准输入，在终端屏幕上输出数据称为标准输出。C的标准I/O库提供了用于数据输入和输出的标准函数。

1.2.4.1 读字符函数getchar

函数getchar读标准输入一个字符，并把这个字符作为函数的返回值。其调用形式如下：

```
char ch;
```

```
ch = getchar( );
```

1.2.4.2 写字符函数 putchar

调用函数 putchar 时，要有字符型参数。该函数把参数字符写入标准输出。其调用形式如下：

```
char ch
……
putchar(ch);
```

下面程序为读标准输入一个字符。如果该字符是小写字母，则转换成相应的大写字母，写入标准输出；否则，不需转换，直接写入标准输出。

```
main()
{
    char c;
    c = getchar();
    if(c >= 'a' && c <= 'z')
        c += 'A' - 'a';
    putchar (c);
}
```

1.2.4.3 格式输出函数 printf

函数 printf 按照格式参数所规定的格式，在标准输出上将值参数的值输出。调用该函数时，要提供格式参数。在需要输出值的情况下，还要提供值参数。调用形式如下：

printf (格式参数，值参数 1，值参数 2，…)；

其中，格式参数是一个字符串。在该字符串中，可以包括格式转换说明和除"换行"之外的任意字符。格式转换说明以%开头，至表 1.2 中列出的格式字符结束。每一组格式转换说明都要求一个值参数和它对应。值参数是要输出其值的量。根据值参数的类型和所希望的数据形式，在格式参数中安排适当的格式转换说明。

表 1.2 格式字符及其意义

格式字符	数据形式
d	十进制数
o	八进制数
x	十六进制数
u	无符号十进制数
c	单个字符
s	字符串，以'\0'结束或满足精度说明的长度为止
f	float 或 double 类型数的一般形式，若不指明精度，小数点后为 6 位数字
e	同 f，用"科学记数法"的形式，通常为[-]d.ddddddE±dd，其中 d 是一位十进制数字

格式转换说明的字域和精度出现在"%"符号和格式字符之间，用负号、数字和小数点表示。其意义如下：

负号 使数据在其域内向左边对齐。

数字 指出最小域宽，必要时还可增加宽度。若实际需要小于最小域宽，则向右对齐，左边留空白；若用负号指出向左对齐，则按多余的域宽在右边留出空白；若在数字串中有前置 0，则在空白处填 0。

句点 指出其后的数字串是精度的位数说明，说明小数点后应取数字的位数或有效字符

的位数。

l或L 长整型数。

在格式参数中，除有上述格式转换说明之外，任何其他字符都按其原来的形式输出。还应注意在1.2.2.1节中提到的转义字符，也可用在格式参数中。此外，如果把%作为一个字符输出，要用%%的形式。

程序format.c给出了各种格式说明的实例。format.r是该程序的输出结果。

Section 1.2.4.3 format.c

```
main( )
{
    int i, j;
    char *s, ch;
    double x, y;
    i = 19; j = 12;
    x = 3.14159265; y = 1.53125;
    s = "the larger number is"; ch=':';
    printf("(1)\ti=%4d\tj=%-4d\n", i, j);
    printf("(2)\ti=%04o\tj=%04o\n", i, j);
    printf("(3)\ti=%04x\tj=%04x\n", i, j);
    printf("(4)\t%s %c %d\n", s, ch, i);
    printf("(5)\tstring address=%u\n", s);
    printf("(6)\tx=%f\ty/x=%.8f\n", x, y/x);
    s = "hello world";
    printf("(7)\t:%10s:\n", s);
    printf("(8)\t:%20s:\n", s);
    printf("(9)\t:%20.10s:\n", s);
    printf("(10)\t:%-20.10s:\n", s);
}
```

Section 1.2.4.3 format.r

```
(1)     i=   19   j=12
(2)     i=0023    j=0014
(3)     i=0013    j=000c
(4)     the larger number is : 19
(5)     string address=6242
(6)     x=3.141593      y/x=0.48741201
(7)     :hello world:
(8)                :hello world:
(9)                :hello world:
(10)    :hello world:
```

函数fprintf为只写终端屏幕函数，通常用作写提示符和诊断信息。其中的格式参数和上述说明相同，stderr是指定的名称，格式如下：

fprintf(stderr, 格式参数, 值参数, …);

1.2.4.4 格式输入函数 scanf

函数 scanf 按照格式参数规定的格式，从标准输入读数据，并将数据存入地址参数所指定的存储单元中。调用形式如下：

scanf(格式参数，地址参数 1，地址参数 2，…)；

其中，格式参数是一个字符串，格式转换说明以%开头，所用的格式字符与表 1.2 中列出的相同。字域宽度在%和格式字符之间用数字说明。在格式参数中，忽略了空格符、制表符和换行符。格式说明之外的其他字符与输入流中的下一个非空白字符相匹配。

地址参数是一个指针变量(见 1.3 节)或为用一元运算符 & 取变量的地址。它所指向的数据类型要与格式转换说明中所用的格式字符相匹配。

在输入流中，可用空格、制表或换行符作为数据分隔符。当输入流中出现与格式转换说明中的格式字符不匹配的数据时，scanf 执行结束，并把已输入数据的个数作为它的返回值。当再次执行该函数时，从输入流中的下一个数据开始读。

程序 scanf. c 是调用 scanf 读数据的例子。该程序读入数据后，调用 printf 将读入的数据输出。scanf. r 是执行该程序的记录，其中，> 是输入行的提示符。

Section 1.2.4.4 scanf.c

```
main( )
{
    int i;
    float x;
    char name[10];
    printf("> ");
    scanf("%d %f %s", &i, &x, name);
    printf("i=%d\tx=%f\tname=%s\n", i, x, name);
}
```

Section 1.2.4.4 scanf.r

```
> 123 4.56 Harbin
i=123   x=4.560000    name=Harbin
```

1.2.4.5 I/O 重定向

在执行程序时从终端输入数据可改成从文件输入数据，把输出数据写入终端也可改成写入文件，这叫 I/O 重定向。如果希望在程序运行时把写标准输出的数据保存在一个文件里，我们可以借助操作系统，给定欲存放数据的文件名，进行输出重定向。例如，prog 是程序的可执行文件的文件名，outfile 是准备存放输出数据的文件名，用命令行

prog > outfile

把本来写入标准输出的信息写入文件 outfile。本书用文件名中带有"·r"的文件来记录程序的输出结果。

反之，也可以把输入数据事先存放在一个文件里，在程序执行时从文件里读数据，从而代替标准输入，进行输入重定向。例如，infile 是存放输入数据的文件，用命令行

```
prog < infile
```

使程序 prog 在执行时，从文件 infile 里读数据。本书用文件名中带有"·d"的文件来存放输入数据。

当然，也可以同时对一个程序进行输入和输出重定向，如命令行

```
prog < infile > outfile
```

使程序 prog 在执行时，从文件 infile 里读数据，把输出结果写入文件 outfile。

还可以把程序的标准输出写在一个文件的末尾，用如下的命令行

```
prog >> outfile
```

这样，不破坏文件 outfile 原来的内容。

1.2.5 条件语句

条件语句的一般形式为：

```
if (表达式)
    语句
else if (表达式)
    语句
...
else
    语句;
```

其中

```
else if (表达式)
    语句
```

的个数不限，也可以没有；并且

```
else
    语句
```

也可以没有。

先求第一个表达式的值，若为 0，再求下一个表达式的值。如果某一个表达式的值不为 0，则执行其后相应的语句，然后，整个 if 语句执行结束。若各表达式的值均为 0，则执行最后一个 else 之后的语句。

在上述条件选择语句中，如果嵌套另外的条件选择语句，要注意每个 else 总和前面最近的 if 相结合。

1.2.6 循环语句

C 有三种循环语句，分别是 while、for 和 do-while 语句。

1.2.6.1 while 语句

一般形式

```
while (表达式)
    语句
```

计算表达式的值，如果不为 0，则执行后面的语句；重复上述过程，当表达式的值为 0 时，

循环结束。

1.2.6.2 for 语句

一般形式

```
for (表达式1;表达式2;表达式3)
    语句
```

如果用 while 语句,其等价形式为:

```
表达式1;
while (表达式2){
    语句
    表达式 3
}
```

通常,表达式1和表达式3都是赋值表达式,分别为对循环变量赋初值和修改循环控制变量。表达式2是循环的条件,若不满足该条件,则循环结束。

在程序 atoi.c 中,函数 atoi 把字符形式的数字串转换成相应的数值。在转换过程中,第一个 for 循环的执行条件是在字符串中有空白,制表或换行符。一旦字符 s[i] 不是上述这些字符之一,循环即结束。于是,下标 i 的值设置在第一个可见字符的位置。第二个 for 循环的执行条件是所遇到的 s[i]都是数字,一旦遇到非数字字符,循环即结束。

Section 1.2.6.2 atoi.c

```
main( )
{
    int x1, x2, x3;
    x1 = atoi(" 1234");
    x2 = atoi(" -234");
    printf("%d + (%d) = %d\n", x1, x2, x1+x2);
}
atoi(s)
char s[ ];
{
    int i, n, sign;
    for (i = 0; s[i] == ' ' || s[i] == '\n' || s[i] == '\t'; i++)
        ;                   /* skip white space */
    sign = 1;
    if(s[i] == '+' || s[i] == '-')
        sign = (s[i++] == '+') ? 1 : -1;
    for(n = 0;s[i] >= '0' && s[i] <= '9'; i++)
        n = 10 * n+ s[i] - '0';
    return(sign * n);
}
```

在程序 reverse.c 中,函数 reverse把字符串的n个字符排列顺序颠倒过来。在循环过程中,因为并行处理下标 i 和 j, 所以, 置初值和修改控制变量要并列进行。在 for 语句里, 用逗号分隔两个表达式,表示并列运算。