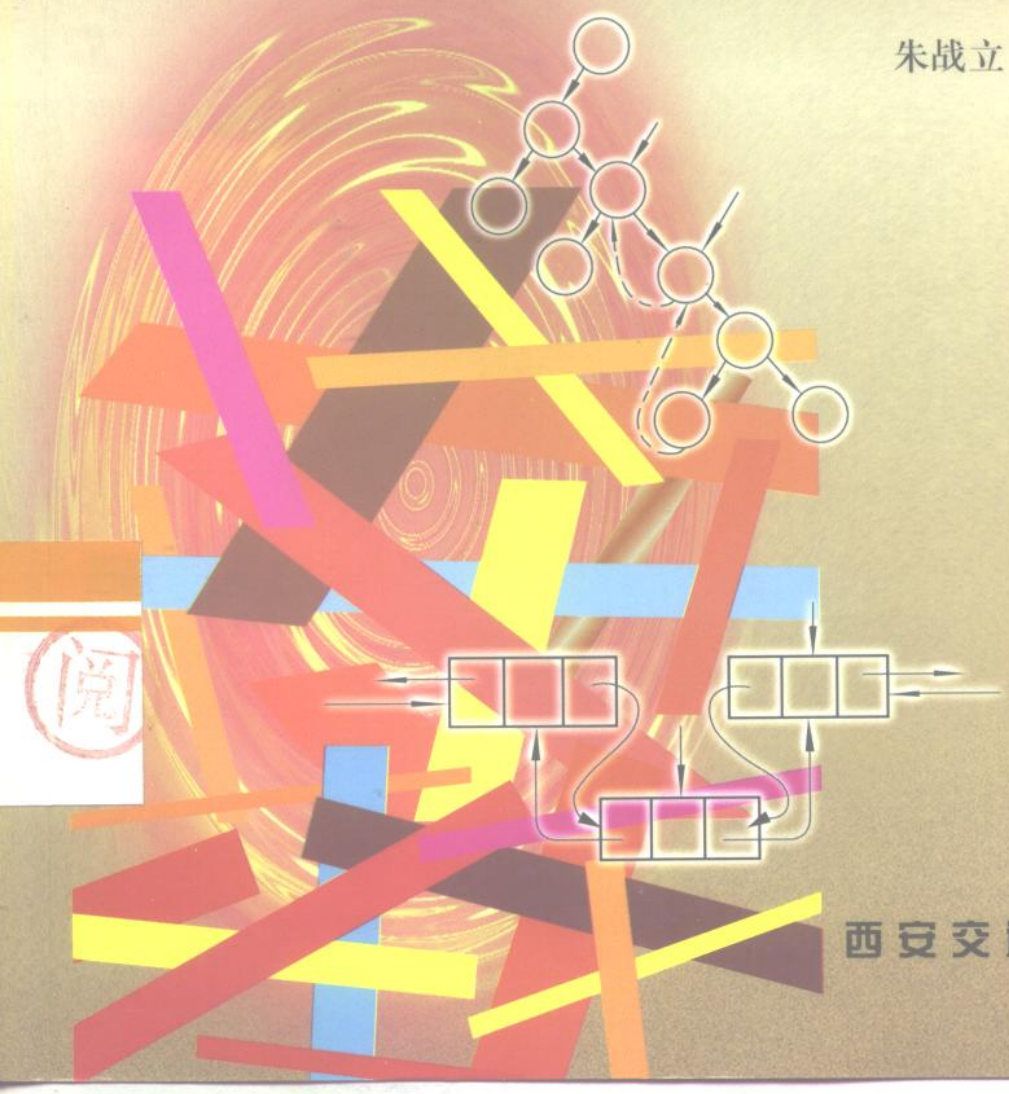
数据结构

——使用C语言

(第2版)

朱战立 刘天时 编著



西安交通大学出版社

数 据 结 构

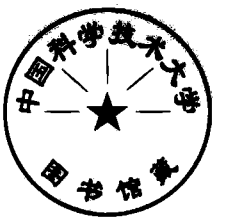
SHUJU JIEGOU

——使用 C 语言

——SHIYONG C YUYAN

（第 2 版）

朱战立 刘天时 编著



西安交通大学出版社

内 容 简 介

本书系统地介绍了各种类型的数据结构和查找、排序的方法。对于每一种类型的数据结构，都详细阐述了基本概念、各种不同的存储结构和不同存储结构上的一些操作实现算法，并列举了许多具体实例帮助读者理解。另外，书中还介绍了递归程序设计方法和文件的组织方法。全书采用C语言作为算法描述语言。

本书既可作为大专院校计算机应用专业和非计算机专业的教科书，也可作为从事计算机应用的工程技术人员的自学参考书。

图书在版编目(CIP)数据

数据结构：使用C语言/朱战立，刘天时编著. —2版.
西安：西安交通大学出版社，2000.2
ISBN 7-5605-0883-9

Ⅰ.数… Ⅱ.①朱… ②刘… Ⅲ.①数据结构 ②算法理论 Ⅳ.TP311.12

中国版本图书馆CIP数据核字(2000)第13090号

出版发行：西安交通大学出版社
（西安市咸宁西路28号 邮政编码：710049）
（电话：(029)2668316）
各地新华书店经销
印刷装订：蓝田县立新印刷厂

开本：787 mm×1092 mm 1/16
印张：21.25
字数：513千字

版次印次：2000年2月第2版 2000年3月第7次印刷
印数：23001～28000
定价：25.00元

第 2 版前言

数据结构是计算机学科本科生、专科生的核心课程，是计算机程序设计的重要理论技术基础。

本书是第 2 版。较之第 1 版，其主要的改动包括：重写了全部算法并规范了算法的书写格式，改写和补充了各章的应用实例，给出了书写习题的部分答案，添加了若干上机实验题目的分析和程序设计。

本书用 C 语言作为算法描述语言。目前，数据结构教材采用 C 语言作为算法描述语言已渐成趋势。数据结构课程对学生编程能力的训练包括两方面的内容：一个方面是训练学生理解各种基本数据结构的概念，并能理解和编写出各种典型的算法；另一个方面是训练学生应用各种典型算法进行具体应用问题的程序设计，这包括程序中变量设计、函数中参数设计、程序的书写格式等方面的训练。目前数据结构教材采用 C 语言作为算法描述语言的也有两种：一种是采用类 C 语言，另一种是直接采用 C 语言。采用类 C 语言有利于学生把注意力放在算法的设计和分析上；采用 C 语言能兼顾两个方面的训练，这对高等工科院校强调对学生实际动手能力的训练是非常适宜的。

第 2 版在附录中添加了部分书写习题答案和若干上机实验题目的分析及程序设计。添加书写习题答案有利于读者自学，添加上机实验题目的分析和程序设计既有利于学生实验报告的书写和实际编程能力的训练，也有利于教师安排和设计学生的上机实验题目。

本书讲授学时可为 44～60 学时。讲授时间低于 60 学时时可适当删去带“*”的章节内容。

本书是在第 1 版基础上，由刘天时完成第 2,3 章和 4.5 节全部算法的改写和完善工作，罗进元重新调试了第 7,8,10 章的几个算法，其余章节的算法改写工作和全部章节的文字修改工作均由朱战立完成，全书由朱战立修改定稿。

编著者

1999.10

前　言

数据结构是一门计算机专业和计算机应用专业本科、专科学生必修的专业基础课。它是许多涉及数据结构设计和算法设计课程，如操作系统、编译原理、计算机图形学、人工智能等的先导课。数据结构也是非计算机专业培养学生软件设计能力重点选择的一门专业课。

本书共分11章。第1章介绍了数据结构的基本概念，概述了C语言的数据类型，说明了算法设计目标和算法效率度量；第2章到第5章分别介绍了线性结构中线性表、堆栈、队列、串和数组的基本概念，以及存储结构和一些基本操作的实现；第6章介绍了递归概念以及递归算法的设计方法和用非递归算法模拟递归算法的基本方法；第7章和第8章分别介绍了非线性结构中树和二叉树以及图的基本概念、存储结构、一些基本操作的实现及它们的典型应用；第9章和第10章分别介绍了排序和查找的各种常用算法，并对各种算法进行了简单的性能分析；第11章介绍了文件的概念和文件的几种组织方法。

本书在内容组织上力求丰富充实，结合实际；在语言描述上力求深入浅出、简洁明了。为了便于学习和理解，书中列举了大量例题，并在一些章后设有专门小节讨论较大型的综合应用问题，每章都配有适量习题。

本书既可作为计算机应用专业本科、专科学生的教材，也可作为非计算机专业本科学生的教材。对从事计算机应用的工程技术人员，本书也是十分有价值的参考书。

目前国内数据结构教材基本使用类PASCAL语言或用PASCAL语言作为算法描述语言，这与C语言的广泛使用和计算机应用专业教学的发展变化要求极不相符。本书是编著者在多年来讲授数据结构和C语言程序设计课程的基础上编写的。本书采用C语言作为算法描述语言，书中全部算法都通过了上机调试。本书第1章至第6章及第11章由朱战立编写，第7章至第10章由李文编写，全书由朱战立修改定稿。

本书由西安交通大学冯博琴教授审阅，西安交通大学出版社对本书的出版给予了大力支持，在此一并表示谢意。

编著者

1997.1

目　　录

第1章 绪 论

计算机是对各种各样数据进行处理的机器。在计算机中如何组织数据，如何处理数据，从而如何更好地利用数据是计算机科学的基本研究内容。掌握数据在计算机中的各种组织和处理方法是继续深入学习的基础。

1.1 数据结构的基本概念

数据结构(Data Structures)研究的是计算机所处理的数据元素间的结构关系及其操作实现的算法。

早期的计算机都用于进行数值计算，数值计算的特点是数据元素间的关系简单，但计算复杂。随着计算机应用范围的扩展，目前计算机更多地被用于非数值处理，如管理、控制等领域，非数值处理问题的特点是数据元素间的关系复杂，而计算较简单。

数据元素间的结构关系(或称逻辑结构)有几种基本形式。最简单的是线性结构，这时其有关结构的性质可以归纳为下述一些问题：哪一个数据元素是线性表中的第一个数据元素？哪一个数据元素是线性表中的最后一个数据元素？某一个数据元素的前驱和后继各是哪一个数据元素等等。更为复杂的结构关系有树形结构——表示着等级和分枝的关系，还有图形结构——表示着更复杂的客观事物之间的关系。

为了有效地在计算机上解决具有各种结构关系的实际问题，我们还必须研究这种具有结构关系的数据在计算机内部的存储方法(或称存储结构)以及在计算机中处理这样的具有结构关系数据所需进行的操作和操作的实现方法。

我们先来看一个最简单的线性表关系的例子。

例 1-1 建立一个住院病人押金情况表。住院病人押金情况表包括：姓名、性别、年龄、住院押金。要求：每当病人住院时，插入一条记录；每当病人出院时，删去该病人的记录。

分析 每个病人的数据为一条记录，每个记录包括姓名、性别、年龄、住院押金四个数据项。所有病人的记录构成一个线性表。这构成了该问题的逻辑数据结构。

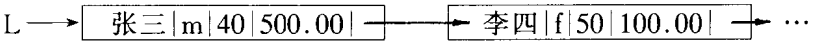
线性表上要求有插入、删除一条记录的操作。插入操作和删除操作构成了该线性表的操作集合。

线性表可以选用不同的存储结构。具体到这个线性表可以有两种存储结构：

(1) 顺序存储结构：即把所有记录依次存储在一个数组中。如

数组下标	姓名	性别	年龄	住院押金
0	张三	m	40	500.00
1	李四	f	50	100.00
⋮	⋮	⋮	⋮	⋮

(2) 单链存储结构:即把每个记录看作一个结点,让所有结点依次链接。如



显然,存储结构不同则实现操作的算法就不同。

由此可见,一个具体问题的软件设计通常包括三个步骤:(1)分析和确定该问题的逻辑数据结构;(2)设计该问题的具体存储结构;(3)设计该问题在具体存储结构下的操作实现算法。

下边我们较为准确地定义我们所引进的几个基本术语:

数据 在计算机中这个术语含义非常广泛,可以认为它是描述客观事物的数字、字符以及所有能输入到计算机中并能为计算机所接受的符号集。

数据元素 数据的基本单位,是计算机进行输入输出操作的最小单位。大多数情况下,一个数据元素由若干个数据项组成。数据项是数据不可分割的最小单位。

数据结构 是指计算机处理的数据元素的组织形式和相互间关系。从数据元素之间的不同特性分,数据结构有三种基本类型:线性结构、树形结构和图形结构。从不同的抽象层次看,数据结构可分为逻辑数据结构和物理数据结构(或称存储结构)。存储结构主要有两种基本类型:顺序存储结构和链式存储结构。

操作 是指数据结构上需要或能够进行的处理。对一个具体问题来说,存储结构不同则操作的具体实现算法就不同;相同的存储下也可有许多种不同的操作实现算法。

我们研究数据结构,需要研究如下问题:一个具体问题的逻辑数据结构是什么?适宜选用什么样的存储结构?采用什么样的操作实现算法效率更高?由此我们可以说,数据结构是一门研究非数值处理问题的软件设计中计算机的操作对象以及它们之间的关系和操作实现等的学科。换句话说,数据结构的研究包括三个方面的内容:数据的逻辑结构、数据的物理存储结构和数据的操作实现算法。

1.2 数据类型和抽象数据类型

1. 数据类型

数据类型这个概念最早出现在高级程序设计语言的实现中。我们知道,计算机硬件系统并不能直接处理如整数、浮点数、双精度数等,需按一定规则转换成二进制码来表示和处理。从计算机硬件系统的角度看,计算机能处理的数据类型只包括位、字节和字。高级程序设计语言的设计者们引入整数、浮点数、双精度数等,对计算机硬件系统来说就是一种新的数据结构。对这种数据结构的操作,需通过编译器转化成机器语言的位、字节和字这样的计算机硬件能接受的数据类型的操作来实现。从高级程序设计语言用户的角度看,数据结构实现了信息的隐蔽,即将一切用户不必了解的细节都封装在数据结构之中,从而实现了数据的抽象。例如,硬件系统直接支持的机器语言程序需考虑整数的具体机器表示和整数运算的机器实现算法,而高级程序设计语言用户在使用整数时,就不需要了解整数在计算机内部是如何表示和各种运算是如何实现的。如对两整数求和这样的问题,高级程序设计语言用户只需了解其数学上求和的抽象操作含义。这样,在硬件系统机器语言的数据类型支持下,高级程序设计语言实现了整数这样的数据结构。进一步,对高级程序设计语言来说,整数就是它的数据类型,或称固有数据类型,高级程序设计语言用户可直接使用。在用高级程序设计语言编写的程序中,每个变量、常量或表达式都有一个它所属的确定的数据类型。数据类型显式或隐式地规定了在程序

执行期间变量、常量或表达式所有可能的取值范围,以及允许进行的操作。例如,C语言中的整数类型,取值范围为[- maxint, maxint]上的整数(maxint 为特定计算机所允许的最大整数),定义在该取值区间上的一组操作为:加(+)、减(-)、乘(*)、整除(/)和取余(%)。

因此,我们可以给数据类型作如下定义:数据类型是一个数据的集合和定义在这个数据集合上的一组操作的总称。或者说,数据类型是某种程序设计语言中已实现的数据结构。

在程序设计语言提供的数据类型的支持下,我们就可根据从问题中抽象出来的各种数据模型,逐步构造出描述这些数学模型的各种新的数据结构。

2. 抽象数据类型

抽象数据类型(Abstract Data Type, 简写为 ADT)是用户在数据类型基础上新定义的数据类型。抽象数据类型定义包括数据组成和对数据的处理操作。可见抽象数据类型是数据和数据的使用者的一个接口。

程序设计方法学的研究指出,一个软件系统的框架应建立在抽象数据类型之上。从抽象数据类型的角度入手设计软件系统可大大提高软件构件的复用率。

抽象数据类型的概念非常类同于面向对象程序设计中类的概念。从抽象数据类型的角度入手设计软件系统使用 C++ 语言较合适,因为 C++ 语言支持面向对象程序设计的类定义。但考虑到和先导课程的衔接,本课程以 C 语言作算法描述语言。

抽象数据类型的定义主要包括数据对象定义、数据关系定义和基本操作定义三部分,其书写格式为:

ADT: 抽象数据类型名

数据对象: 数据对象的定义

数据关系: 数据逻辑关系的定义

基本操作: 基本操作的定义

其中数据对象定义应是在已有数据类型或已定义数据对象基础上对新的数据对象作出定义,基本操作定义主要包括操作名、参数表、初始条件和操作结果四部分内容的定义和描述,其书写格式为:

操作名(参数表)

操作结果: 操作结果描述

由于 C 语言只能非常有限地支持抽象数据类型概念,因此本书未从抽象数据类型的角度出发进行讨论,但在有些地方给出了 C 语言支持的最简单的抽象数据类型的实现方法。

1.3 C 语言的数据类型

本节我们简要回顾 C 语言的数据类型。

1. C 语言的基本数据类型

C 语言有三种由硬件系统直接支持实现的基本数据类型:int 型、float 型和 char 型。int 型可以有三个限定词:short, long 和 unsigned。short 或 long 缩小或扩大了 int 型整数的存储空间和数值表示范围。但精确的 short int, int 和 long int 型存储空间和数值表示范围由具体 C 版本和具体机器确定。unsigned int 表示其变量只取正整数。

float 型有三种形式:float, double 和 long double,其存储空间分别为 4 个字节、8 个字节和

16 个字节。

char 型存储空间为一个字符。由于字符在机器内以整数形式表示，所以 C 语言也允许用 char 型定义占 1 个字节、其数值范围为 -127～128 的变量，但是为避免混淆，本书将不这样使用。

2. C 语言的指针类型

C 语言允许直接对存放变量的地址进行操作。如定义 int a，则 &a 表示变量 a 的地址，也称作指向变量 a 的指针。存放地址的变量称作指针变量。如再定义 int *pa，则语句 pa = &a 即为把变量 a 的地址赋给指针变量 pa；而语句 a = *pa 则为把指针 pa 所指地址中的值赋给变量 a。另外，这里 pa 的含义不只是指针，而且是确定的整型类型指针。这是因为不同数据类型变量在内存中分配的空间大小不同。假设 int 型占 isize 个字节，并设系统为 pa 分配的内存地址为 1000，则 pa + 1 为 1000 + isize；而 *(pa + 1)表示从 1000 + isize 开始的 isize 个字节中的整型数。

C 语言中指针的一个重要作用是在函数间传递数据。对一个形参为简单变量的函数，参数传递是把实参的值拷贝给形参，即通常说的值传递。这时，函数内形参的值被修改时实参的值不会跟着变化，这适用于函数参数为输入参数的情况。若函数采用指针作形参，参数传递仍是把实参的值(是一个地址)拷贝给形参，即通常说的地址传送，若函数内形参的指针值改变，则实参中的指针值将跟着变化。利用指针类型形参可设计函数的非指针类型的输出参数和变参。

例 1-2 值传送和地址传送。

(1) 值传送

```
main()
{
    int x = 5;
    printf("\n%d", x);
    Function1(x);
    printf("\n%d", x);
}

void Function1(int x)
{
    x++;
    printf("\n%d", x);
}
```

输出结果为：

5(第 2 行的输出)

6(函数中的输出)

5(第 4 行的输出，不随形参改变)

(2) 地址传送

```
main()
{
    int x=5;
    printf ("\n%d", x);
    Function2(&x);
    printf ("\n%d",x);
}

void Function2 (int *px)
{
    (*px)++;
    printf ("\n%d", *px);
}
```

输出结果为：

5(第2行的输出)

6(函数中的输出)

6(第4行的输出,随着形参一起变化)

对于函数的指针类型的输出参数和变参,设计方法可依此类推,即可设计成指针的指针类型(具体例子请参阅2.5.1节例2-5的算法3)。

3. C语言的数组类型

数组是同一类型的一组数据的集合。数组名标识了这一组数,数组元素下标指示了数组元素在该数组中的位置。可以有一维数组和多维数组。数组的两个最基本操作是存和取。假如有定义 int a[100],x;则语句 a[i]=x 表示把 x 的值赋给 a[i];语句 x=a[i]表示把a[i]的值赋给 x。

数组下标的最小值称为下界,这在C语言中总是0。数组下标的最大值称为上界,C语言中数组上界为数组定义值减1,如上例数组 a 的上界即为99。数组的上下界范围等于上界减下界加1,如上例数组 a 的上下界范围为100。

一个数组名代表了这一组数的起始地址,也就是说,数组名是一个指针变量。若 a 是如上定义的一维数组,则有:

$$\&a[0]=a,\quad \&a[1]=a+1,\ \cdots,\ \&a[i]=a+i$$

$$a[0]=*a,\quad a[1]=*(a+1),\ \cdots,\ a[i]=*(a+i)$$

假如还有定义 int *p,则语句 p=a 表示把数组 a 的首地址赋给指针变量 p。语句 p+1 等价于语句 a+1,但 p 是指针变量,本身可改变,a 是地址常量,本身不可改变。

当数组作为函数参数时,其传递方法和指针作为函数参数的传递方法相同,即采用地址传送法。因此,对函数中的一维数组形参,函数定义时仅需定义形参数组名,而不必定义数组上下界范围,因为系统并不实际为形参数组分配内存。函数中的数组形参也可用指针变量表示,因为当无需定义数组范围时,数组形参名和指针变量都是指针类型,两者没有差别。

由于数组作为函数参数时实参和形参的传递方式是地址传递法,因此形参中数组的值的

改变可传递给实参。

例 1-3 设计一个程序，将原数组中的值转换为逆序，即设原数组中的值为(a_0, a_1, …, a_{n-1})，逆序则为(a_{n-1}, …, a_1, a_0)。要求把转换算法设计成函数。

程序和函数设计如下：

```
main()
{
    int a[] = {1,2,3,4,5,6,7,8}, i;
    void Converse(int a[], int n);

    printf("\n 原数组序列为:");
    for(i=0; i<8; i++) printf("%d ",a[i]);

    Converse(a,8);

    printf("\n 逆序数组序列为:");
    for(i=0; i<8; i++) printf("%d ",a[i]);
}

void Converse(int a[], int n)
/* 把有 n 个元素的数组 a 逆序存储 */
{
    int m, i, x;

    m = n/2;
    for(i=0; i<m; i++)
    {
        x = a[i];
        a[i] = a[n-1-i];
        a[n-1-i] = x;
    }
}
```

运行结果如下：

```
原数组序列为:
1  2  3  4  5  6  7  8
逆序数组序列为:
8  7  6  5  4  3  2  1
```

可见，形参中数组值的改变传递到了实参中。

一个二维数组可看作其每一个数组元素都是一个一维数组的一维数组。假如有定义

int b[3][4];

则要存取二维数组 b 需要给出行下标(第一维)和列下标(第二维)。但这实际上是一个逻辑数据结构,因为计算机的内存是一维的,只能一个一个元素顺序存放。C 语言中其物理存储结构是以行主序存放方式实现的,即第一行的所有元素存完后,再顺序存放第二行的元素,依此类推。假设二维数组 b 的起始地址为 base(b),整型类型的字节数为 isize,则二维整型数组元素 b[i][j](0≤i<3,0≤j<4)的内存地址为:

base(b) + (i * 4 + j) * isize

由此可知,二维数组有两个指针,一个是行指针,一个是列指针。二维数组 b 有图 1-1 所示的指针关系。

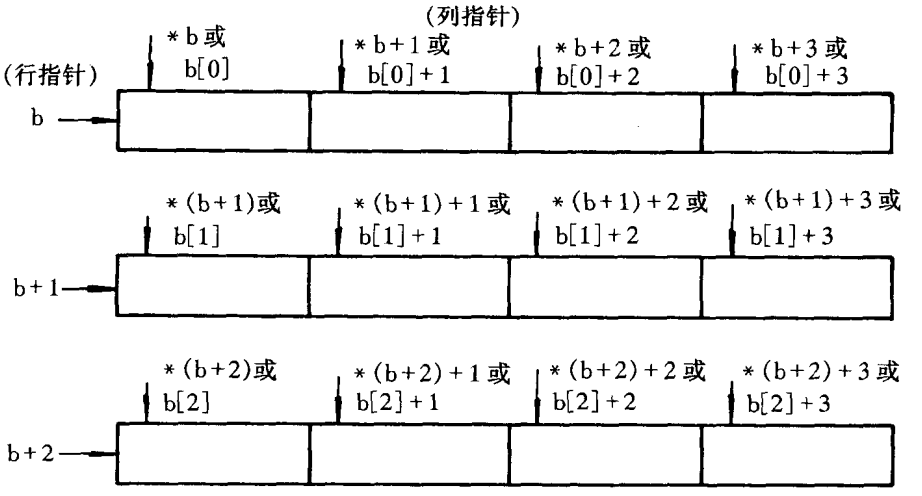


图 1-1 二维数组的行指针和列指针

其中行指针加 1 表示加 4 * isize 个字节,列指针加 1 表示加 isize 个字节。

多维数组的定义方法和内存存储顺序可以类推。

C 语言中编译系统并不检查数组下标是否越界,需要用户自己注意。另外,C 语言不允许数组整体进行存取操作。

4. C 语言的字符串

C 语言中没有单独的字符串类型,字符串定义成字符数组。每个字符串由转义符'\0'指示其结束。一个字符串常量由一对双引号指示。一个字符串常量被存入内存时,系统自动在其末尾添加字符串结束标志'\0'。假设有如下初始化定义:

char string [80] = "Data structure";

则 char 型的一维数组 string 的数组范围是 80,字符串长度是 14。字符串长度不包括'\0'字符,但'\0'字符需占用一个字符空间。因此字符串必须满足:字符串长度≤数组范围 - 1。

C 语言中对字符串的操作提供有函数,如字符串长度函数 strlen(string),字符串的拷贝函数 strcpy(str1,str2),字符串的连接函数 strcat(str1,str2),等等。这些函数都包含在头文件 string.h 中。

5. C 语言的结构体类型

结构体由一组称为结构体成员的项组成,每个结构体成员都有自己的标识符。在有些高