普通高等教育计算机类专业“十三五”规划教材

C语言与程序设计

(第2版)

主　编　胡元义　王　磊
副主编　吕林涛　高　勇　崔俊凯　谈姝辰　鲁晓锋

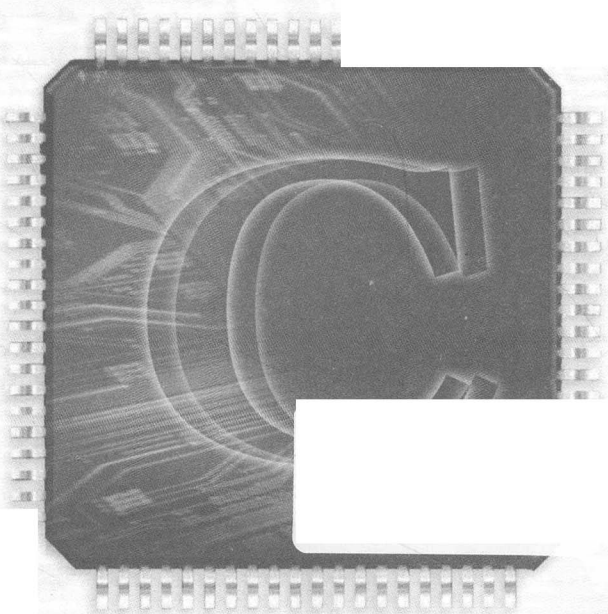
西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

普通高等教育计算机类专业“十三五”规划教材

C语言与程序设计

(第2版)

主　编　胡元义　王　磊
副主编　吕林涛　高　勇　崔俊凯　谈姝辰　鲁晓锋



西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

内容简介

本书作为程序设计课程的教材，在结构上注重知识的系统性、完整性和连贯性，将理论与实践有机结合。作者在总结多年教学与实践的基础上，精选了约400道设计独到的例题来作为典型概念示例和程序精讲，并且兼顾C语言等级考试，所有程序例题与习题都在VC++6.0环境下上机通过。对重点章节如函数和指针内容，作者采用了独创的动态图分析方法来分析程序执行中函数或指针的变化情况，使函数和指针内容中难以掌握的部分迎刃而解。本书编写循序渐进、深入浅出且图文并茂，力求达到使读者深入掌握C语言程序设计的目的。

本书除了可以作为程序设计语言教材外，还可以作为全国计算机等级考试的教材或参考书。对于从事计算机专业的工作者，本书也是一本难得的资料书。

图书在版编目(CIP)数据

C语言与程序设计/胡元义，王磊主编. —2版. —
西安：西安交通大学出版社，2017.8
ISBN 978-7-5605-9868-0

Ⅰ.①C… Ⅱ.①胡… ②王… Ⅲ.①C语言-程序设计-教材 Ⅳ.①TP312.8

中国版本图书馆CIP数据核字(2017)第168506号

书　　名　C语言与程序设计(第2版)
主　　编　胡元义　王　磊
副 主 编　吕林涛　高　勇　崔俊凯　谈姝辰　鲁晓锋
责任编辑　王　欣

出版发行　西安交通大学出版社
（西安市兴庆南路10号　邮政编码710049）
网　　址　http://www.xjtupress.com
电　　话　(029)82668357　82667874(发行中心)
(029)82668315(总编办)
传　　真　(029)82668280
印　　刷　西安建科印务有限责任公司

开　　本　787mm×1092mm　1/16　**印张**　23.25　**字数**　565千字
版次印次　2017年8月第2版　2017年8月第1次印刷
书　　号　ISBN 978-7-5605-9868-0
定　　价　46.50元

读者购书、书店添货、如发现印装质量问题，请与本社发行中心联系、调换。
订购热线：(029)82665248　(029)82665249
投稿热线：(029)82664954
读者信箱：jdlgy@yahoo.cn

第 2 版前言

本书作为程序设计课程的教材，在结构上注重知识的系统性、完整性和连贯性；在内容上突出重点，分散难点；在讲授中循序渐进、深入浅出，将理论与实践有机结合，融知识传授与能力培养于一体。

作者在总结多年教学与实践的基础上，精选了大量内容生动、设计独到的例题来作为典型概念示例和程序精讲，并且兼顾 C 语言等级考试，许多例题就是选自历年二级 C 语言等级考题试题。全书给出了近 400 道例题，且所有程序例题与习题都在 VC++6.0 环境下上机通过。本书在例题分析中大量采用了图示说明，这样使解题思路更加一目了然。对重点章节如函数和指针内容，作者采用了独创的动态图分析方法来分析程序执行中函数或指针变化的情况，使这些难点更容易被读者理解。此外，对采用指针来指向数组元素的相关内容，作者采用了新颖的表述方法来解决同一个数组元素有多种表示法的问题。对于文件的讲解，作者也辅以图片来进行说明，以便读者能够深入了解文件内部的读写过程。

本书第 1 章介绍了计算机的基本知识和程序设计的基本概念，并在此基础上介绍了 C 语言的发展历程和特点，同时还介绍了 C 语言程序的基本组成以及在 Visual C++环境下的上机操作。第 2 章介绍有关 C 语言程序设计的基础知识，如：C 语言的基本符号与基本数据类型，C 语言常量、变量的概念和使用规则，C 语言的运算符与表达式，以及对 C 语言数据的输入和输出方法。第 3 章介绍了如何使用顺序、选择和循环三种基本结构来进行程序设计的方法，这是程序设计最基本的内容，也是真正掌握编程的一个必由之路。第 4 章的数组实际上是一个“量”的扩展，即由对少量的个别数据的处理编程扩展到对大量的成批数据的处理编程，因此引入了存放成批数据的数据结构——数组。第 5 章函数实际上是对“程序结构”的扩展，即程序由一个单一的主函数扩展到多个函数时如何定义和调用这些函数？参数如何在函数之间传递？计算结果又如何由被调函数返回？这些都将在第 5 章里得到解答。第 6 章的指针实际上是对变量访问的扩展，通过指针可以有效地表示各种复杂的数据结构，从而编写出精炼且高效的程序来。第 7 章的结构体是在第 4 章数组简单“量”的扩展基础上的又一个更高层次的扩展，即将不同的简单“量”组合在一起形成一个复杂的“量”——结构体，进而也可以形成一批结构体的“量”。第 8 章介绍了 C 语言程序如何处理来自外存的数据，即如何与

外存文件中的数据打交道。此外，对于那些与各章内容没有紧密联系或无关紧要又较少使用的内容，则统统归于第9章“C语言知识补遗”，这样使各章的知识更为紧凑、清晰和精炼。

本书在章节内容和安排上也进行了调整，第3章至第8章均在最后增加了一节“典型例题精讲”，以利于开拓读者解题思路和提高编程能力，以达到举一反三的目的。本书所讲授的内容均基于VC++环境。

本书配有《C语言与程序设计习题解析及上机指导(第2版)》，可供师生参考。建议两书配合使用，以达到更好的教学效果。本书带“*”的内容为选讲内容，可根据讲授时数进行取舍。

本书除了可以作为程序设计语言教材外，还可以作为全国计算机等级考试的教材或参考书。对于从事计算机专业的工作者，本书也是难得的一本资料书。

欢迎读者对本书的内容及本书中作者的某些见解和表述方法提出批评指正。

编　者

2017年1月于西安

目　录

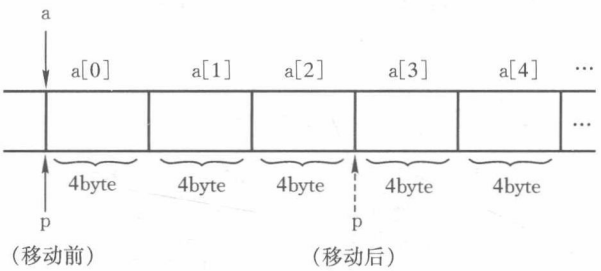


图 6-3 执行"p=p+3;"语句前后 p 指针位置示意

指针变量"++""--"运算也是如此,指针变量"++"运算后指向下一个数组元素地址,而指针变量"--"运算则指向前一个数组元素地址。

此外,还可以给指针变量赋空值"NULL"或 0 值,即该指针变量不指向任何变量(实际上 NULL 和转义字符'\0'都是整数 0,即 NULL 和'\0'的 ASCII 码值均为 0)。例如:

```
int *p;
p=NULL;
```

当两个指针变量指向同一数组时,两个指针变量相减才有意义。相减结果的绝对值表示这两个指针变量之间数组元素的个数。注意,两个指针变量不能做加法运算,因为没有任何实际意义。

(3) 指针变量的关系运算。两个指针变量(必须指向相同类型的变量)之间的关系运算,表示它们指向的变量其地址在内存中的位置关系,即存放地址值大的指针变量大于存放地址值小的指针变量。因此,两个指针变量之间可以进行">""<=""<""<=""==""!="这六种关系的比较运算。

例 6.2 求出下面程序的运行结果。

```
#include<stdio.h>
void main()
{
    int i=10,j=20,k=30;
    int *a=&i,*b=&j,*c=&k;
    *a=*c;
    *c=*a;
    if(a==c)
        a=b;
    printf("a=%d,b=%d,c=%d\n",*a,*b,*c);
}
```

解 需要注意的是,a 表示它所指向的那个变量的地址,而 *a(引用时)则表示它所指向的那个变量的值。我们可以从程序中看出:*a 等于 *c,其值为 30;而 *b 不变,其值为 20。条件语句中的表达式"a==c"判断的是两个地址值,而 a 为 i 的地址,c 为 k 的地址,这两个地址值必然不等,因此赋值语句"a=b;"没有执行。由此得到输出结果如下:

a=30,b=20,c=30

我们也可以用动态图的方法进行分析。由于 a、b、c 为指针变量，它们都是指向其他变量的，因此我们画一个由指针变量到它指向的那个变量的箭头(此后，凡是遇到 * a 这种形式，都是从 a 开始依据箭头找到所指的那个变量)，然后对那个变量进行操作。本题程序执行的动态图见图 6-4，由图 6-4 可以很容易地得到运行结果为

a=30,b=20,c=30

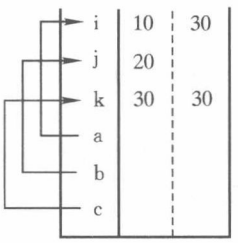


图 6-4　程序执行的动态图

6.2　指针变量与数组

6.2.1　指针变量与一维数组

一个数组在内存中的存储是由一段连续的内存单元组成的，数组名就是这段连续内存单元的首地址。而对数组元素的访问就是通过数组名(即数组的起始位置)加上相对于起始位置的位移量(即下标)来得到要访问数组元素的内存地址，然后对该地址中的内容进行操作。在第 4 章我们已经知道，数组名代表该数组首元素的地址。因此，数组名与我们这里介绍的指针概念相同。实际上，C 语言就是将数组名规定为指针类型的符号常量，即数组名是指向该数组首元素的指针常量(即地址常量)，其值不能改变(即始终指向数组的首元素)。C 语言对数组的处理，也是转换成指针运算完成的。例如：

```
int a[10], * p;
```

则下面的两个语句等价：

```
p=&a[0];
p=a;
```

其作用都是使 p 指向 a 数组的第 0 号元素(即 a 数组的首元素)，见图 6-5。也就是，指针变量的指针值是数组元素的地址，此时有 p 等于 &a[0]和 * p 等于 a[0]。

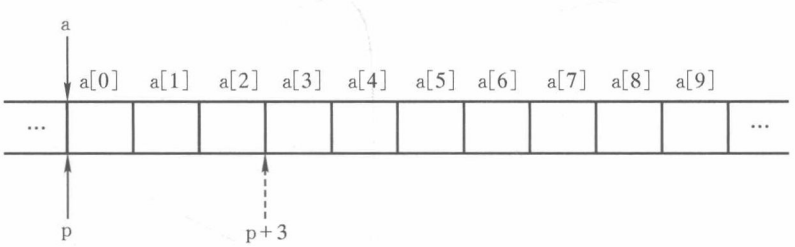


图 6-5　执行“p=a;”后的内存示意

我们知道，a[i]代表 a 数组中的第 i+1 个元素(因为由下标 0 开始)。因此，从图 6-5 可知，p[i]与 a[i]相同，也代表着 a 数组的第 i+1 个元素。

那么 a+1 又代表着什么呢？由 6.1.3 节指针变量的引用和运算可知，这个“1”表示一个数组元素单位，即 a+1 代表第 2 个元素 a[1]的地址。同样，p+1 也代表着 a[1]的地址。因

此,a+i和p+i都代表着第i+1个元素a[i]的地址&a[i]。此外,由"*"运算符可知:*(p+i)和*(a+i)则代表着数组元素a[i]。因此,引用一个数组元素就可以采用以下两种方式:

(1) 下标法:采用a[i]或p[i]的形式访问a数组的第i+1个元素;

(2) 指针法:采用*(a+i)或*(p+i)的形式访问a数组的第i+1个元素。

例6.3 给数组输入10个整型数,然后输出显示。

解 实现方法如下:

(1) 下标法。

①
```
#include<stdio.h>
void main()
{
    int i,a[10];
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    for(i=0;i<10;i++)
        printf("%4d",a[i]);
}
```

②
```
#include<stdio.h>
void main()
{
    int *p,i,a[10];
    p=a;
    for(i=0;i<10;i++)
        scanf("%d",&p[i]);
    for(i=0;i<10;i++)
        printf("%4d",p[i]);
}
```

(2) 指针法。

```
#include<stdio.h>
void main()
{
    int *p,a[10];
    for(p=a;p<a+10;p++)
        scanf("%d",p);
    for(p=a;p<a+10;p++)
        printf("%4d",*p);
}
```

(3) 指针地址位移法。

①
```
#include<stdio.h>
void main()
{
    int i,a[10];
    for(i=0;i<10;i++)
        scanf("%d",a+i);
    for(i=0;i<10;i++)
        printf("%4d",*(a+i));
}
```

②
```
#include<stdio.h>
void main()
{
    int *p,i,a[10];
    p=a;
    for(i=0;i<10;i++)
        scanf("%d",p+i);
    for(i=0;i<10;i++)
        printf("%4d",*(p+i));
}
```

注意,在第(2)种方法指针法中,循环控制表达式中的"p++"使得指针变量p的指向能够

逐个元素的移动，从而实现对每一个数组元素的访问；在输出过程中，当输出完最后一个数组元素 a[9]的值时，p 的指针值已移至 a[9]元素之后的位置(即 a 数组范围之外)。由于数组名 a 是指向该数组首元素的指针常量，它不能实现移动，故指针法只有一种方法。

在数组中采用指针方法应注意以下几点：

(1) 用指针变量访问数组元素时，要随时检查指针变量值的变化，不得超出数组范围。

(2) 指针变量的值可以改变，但数组名的值不能改变。如例 6.3 中，p++正确，而 a++错误。

(3) 对于 *p++，其结合方向为自右至左，因此等价于 *(p++)。

(4) (*p)++表示 p 所指向的数组元素的值加 1，而不是指向其后的下一个数组元素。

(5) 如果当前的 p 指向 a 数组中的第 i+1 个元素，则

*(p++)　等价于　a[i++]

*(p--)　等价于　a[i--]

*(++p)　等价于　a[++i]

*(--p)　等价于　a[--i]

注意：*(p++)与 *(++p)作用不同。若 p 的初值为 &a[0]，则 *(p++)等价于 a[0]，而 *(++p)等价于 a[1]。

(6) 区分下面的指针含义：

① ++*p 相当于++(*p)，即先给 p 指向的数组元素的值加 1，然后再取这个数组元素的值。

② (*p)++则是先取 p 所指数组元素的值，然后给该数组元素的值加 1。

③ *p++相当于 *(p++)，即先取 p 所指数组元素的值，然后 p 加 1 使 p 指向其后的下一个数组元素。

④ *++p 相当于 *(++p)，先使 p 指向其后的下一个数组元素，然后再取 p 所指向的数组元素的值。

例 6.4　给出下面程序的运行结果。

```
#include<stdio.h>
void main()
{
    int a[10]={10,9,8,7,6,5,4,3,2,1},*p=a+4;
    printf("%d\n",*++p);
}
```

解　程序执行示意如图 6-6 所示。首先将 p 定位于(即指向)数组元素 a[4]，在输出时先执行++p(即 p 已指向 a[5]元素)，然后再取该元素的值，故输出结果为 5。

例 6.5　给出下面程序的运行结果。

```
#include<stdio.h>
void main()
{
    int a[]={9,8,7,6,5,4,3,2,1,0};
    int *p=a;
```

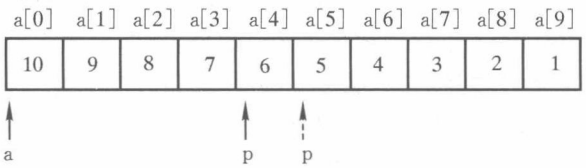


图 6-6 程序执行中 p 指针变化示意

```
    printf("%d\n", *p+7);
}
```

解 由程序可知,p 已指向 a[0],而 *p+7 则是先取出 p 所指向数组元素 a[0]的值,然后再加 7,而 a[0]的原值为 9,故输出结果为 16。

例 6.6 给出下面程序的运行结果。

```
#include<stdio.h>
void main()
{
    int i,a[10]={10,20,30,40,50,60,70,80,90,100}, *p;
    p=a;
    for(i=0;i<10;i++)
        printf("%4d", *p++);
    printf("\n");
    p=a;
    for(i=0;i<10;i++)
        printf("%4d",(*p)++);
    printf("\n");
}
```

解 *p++表示先输出 p 所指向元素的值,然后 p 加 1 指向其后的下一个元素。(*p)++则是先输出 p 所指向的数组元素的值,然后再给这个数组元素值加 1,而 p 指针的指向不变。因此,我们得到输出结果如下:

```
  10   20   30   40   50   60   70   80   90  100
  10   11   12   13   14   15   16   17   18   19
```

6.2.2 指针变量与二维数组

1. 二维数组元素地址及元素的表示方法

由于二维数组是多维数组中比较容易理解的一种,并且它可以代表多维数组处理的一般方法,所以这里主要介绍指针变量和二维数组的关系。

为了说明问题,我们定义一个二维数组如下:

```
int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

由第 4 章可知,数组名 a 是二维数组 a 的起始地址,也就是数组元素 a[0][0]的地址,而 a[0]、a[1]、a[2]则分别代表数组 a 各行的起始地址(见图 6-7)。

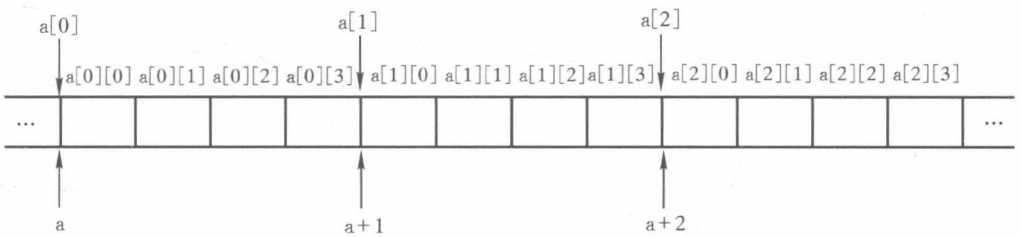


图 6-7　二维数组地址示意

由 6.2.1 节可知，a+i 在一维数组 a 中表示从数组 a 首地址开始向后位移 i 个元素位置，即为一维数组 a 中第 i+1 个元素的地址。在二维数组中，a+i 仍然表示一个地址，但 i 值不再像一维数组那样以元素为单位而是以行为单位了，即将整行看做为一维数组中的一个元素。这样，a+i 就代表二维数组 a 的第 i 行的首地址。因此，在二维数组中，a+i 与 a[i]等价(见图 6-7)。

我们知道，在一维数组中 a[i]与 *(a+i)等价，它们都代表着一维数组 a 中的第 i+1 个元素。而在二维数组中，a[i]不再是数组元素而表示一个地址。因此，在二维数组中与 a[i]等价的 *(a+i)也表示一个地址，即它与 a[i]都代表着二维数组中第 i 行的首地址(*(a+i)本身也无法表示二维数组某行某列的数组元素)。

因此在二维数组 a 中，数组元素 a[i][j]的地址可采用下列形式表示：

```
(1) &a[i][j]          /* 行下标和列下标表示法 */
(2) a[i]+j            /* 行下标加列位移表示法 */
(3) *(a+i)+j          /* 行位移加列位移表示法 */
```

在此，我们一定要注意：在一维数组中 a[i]和 *(a+i)均代表一个数组元素，而在二维数组中它们却代表一个地址。此外，&a[i]也表示二维数组 a 第 i 行的首地址，这样在二维数组的地址中就有 &a[i]、a[i]与 *(a+i)三者等价。

对于二维数组 a，我们知道 a 是指向 a 数组的开始位置，而 *a(即 *(a+0))则是指向 a 数组第 0 行开始位置(即 a[0])，而 **a 才代表着 a 数组第 0 行第 0 列的数组元素 a[0][0]。相应地，数组元素 a[i][0]也可用 **(a+i)表示，即如果要用行位移加列位移表示法来表示一个二维数组元素，则必须经过两次“*”运算才能实现。二维数组中的数组元素 a[i][j]也有如下的三种表示方法：

```
(1) a[i][j]                 /* 行下标和列下标表示法 */
(2) *(a[i]+j)               /* 行下标加列位移表示法 */
(3) *(*(a+i)+j)             /* 行位移加列位移表示法 */
```

显然，一维数组元素和二维数组元素表示的区别(在不含“&”的情况下)是：一维数组元素仅有一个“*”或一个“[ ]”，如 a[i]、*(a+i) 和 *a(即 a[0])；二维数组元素则是“*”和“[ ]”之和的个数必须是 2，如a[i][j]、*(a[i]+j)、*(*(a+i)+j) 和 **a(即 a[0][0])。

例 6.7　用不同方法实现对二维数组的输入和输出。

解　实现方法如下：

(1) 下标法。

```
#include<stdio.h>
```

```
void main()
{
    int a[3][4],i,j;
    for(i=0;i<3;i++)
        for(j=0;j<4;j++)
            scanf("%d",&a[i][j]);
    for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
            printf("%4d",a[i][j]);
        printf("\n");
    }
}
```

(2) 行下标加列位移法。

```
#include<stdio.h>
void main()
{
    int a[3][4],i,j;
    for(i=0;i<3;i++)
        for(j=0;j<4;j++)
            scanf("%d",a[i]+j);
    for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
            printf("%4d",*(a[i]+j));
        printf("\n");
    }
}
```

(3) 行位移加列位移法。

```
#include<stdio.h>
void main()
{
    int a[3][4],i,j;
    for(i=0;i<3;i++)
        for(j=0;j<4;j++)
            scanf("%d",*(a+i)+j);
    for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
```