西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

数组元素的简洁表示

2、指向多维数组元素的指针

7.3.2 指向多维数组元素的指针

- 前面讲到，多维数组可以看作是一维数组概念上的递归延伸，其存储形式也是线性的，即元素的内存单元是连续排列的。本质上，C语言将多维数组当成一维数组来处理。

7.3.2 指向多维数组元素的指针

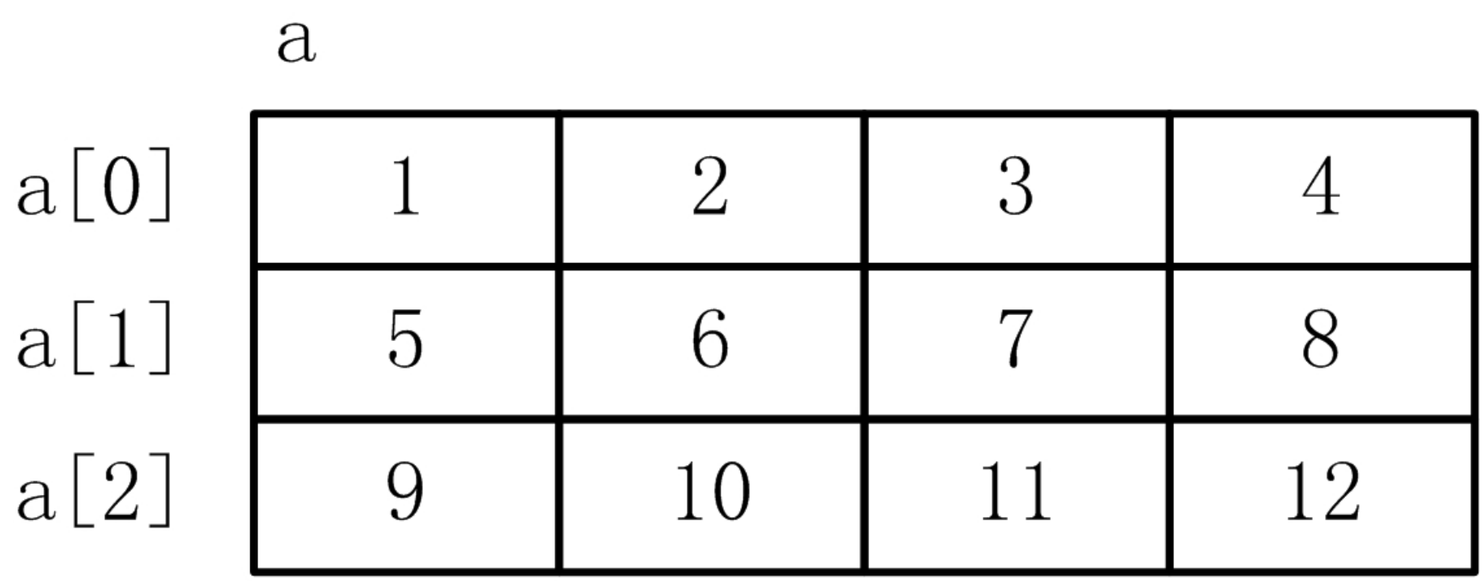
- 1. 多维数组元素的地址
- 以二维数组为例，假设有定义

```
int a[3][4];
```

- 可以将数组a理解为由3个一维数组组成，即a由a[0]、a[1]、a[2]3个元素组成，其中每个元素又是一个一维数组，包含4个元素，例如a[0]（一维数组）有4个元素a[0][0]、a[0][1]、a[0][2]、a[0][3]。如图所示。

7.3.2 指向多维数组元素的指针

图7.12 指向二维数组的指针



（a）

7.3.2 指向多维数组元素的指针

- 二维数组a的12个元素在内存中是连续排列的。数组a先按行排列，即先存放第0行a[0]，其次为第1行a[1]、第2行a[2]。在存放第0行a[0]时，按一维数组形式将它的4个元素一一存放，以此类推，直至数组a所有元素全部存放。显然，N维数组也是这样的规律，即先将最高维当作一个一维数组，将其每个元素一一存放，而每个元素又是一个N-1维数组，递归处理直至数组所有元素全部存放。

a

a[0]	1	2	3	4
a[1]	5	6	7	8
a[2]	9	10	11	12

7.3.2 指向多维数组元素的指针

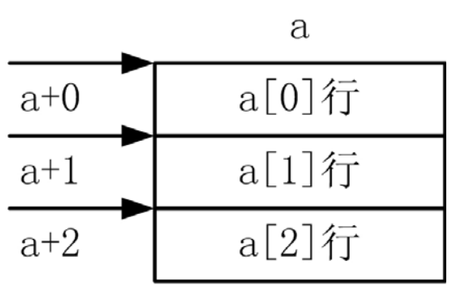
- 为了得到数组a每个元素的地址，可以对数组元素使用取地址运算“&”。例如：
- &a[0][0]是数组元素a[0][0]的地址
- &a[i][j]是数组元素a[i][j]的地址
- 而数组名a既代表数组对象，又是数组的首地址，即a与&a[0][0]等价。

a

a[0]	1	2	3	4
a[1]	5	6	7	8
a[2]	9	10	11	12

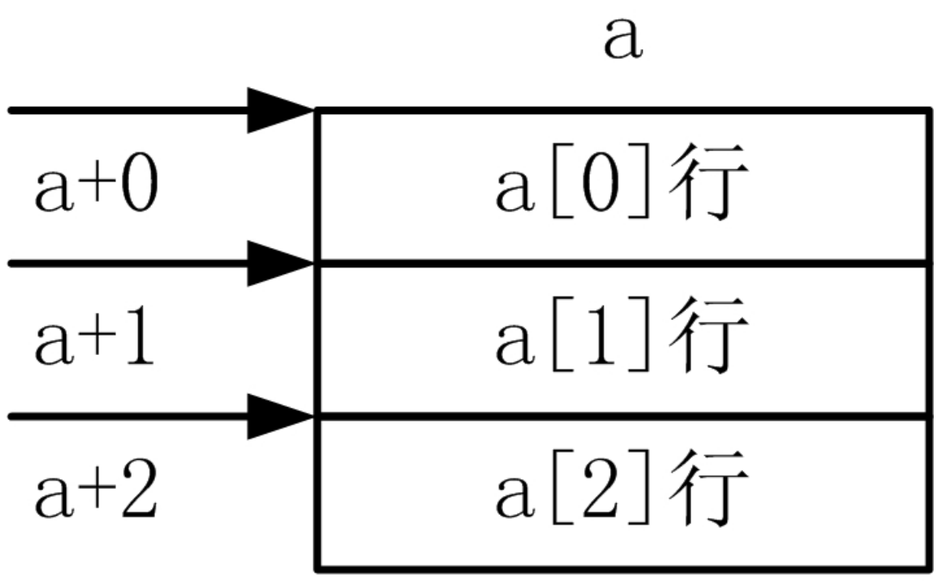
7.3.2 指向多维数组元素的指针

- 从二维数组的角度来看，a是二维数组首元素的地址，而这个首元素不是一个整型元素，而是由4个整型元素所组成的一维数组，因此a+0（即a）是第0行a[0]的首地址，a+1是第1行a[1]的首地址，a+i是第i行a[i]的首地址。需要注意，a和a+1的地址值相差了4*sizeof(int)，因为两行之间间隔了4个整型元素。由于a[i]表示第i行，则&a[i]是第i行的地址，即&a[i]与a+i等价，它们均指向第i行（一个一维数组）。如图所示。



7.3.2 指向多维数组元素的指针

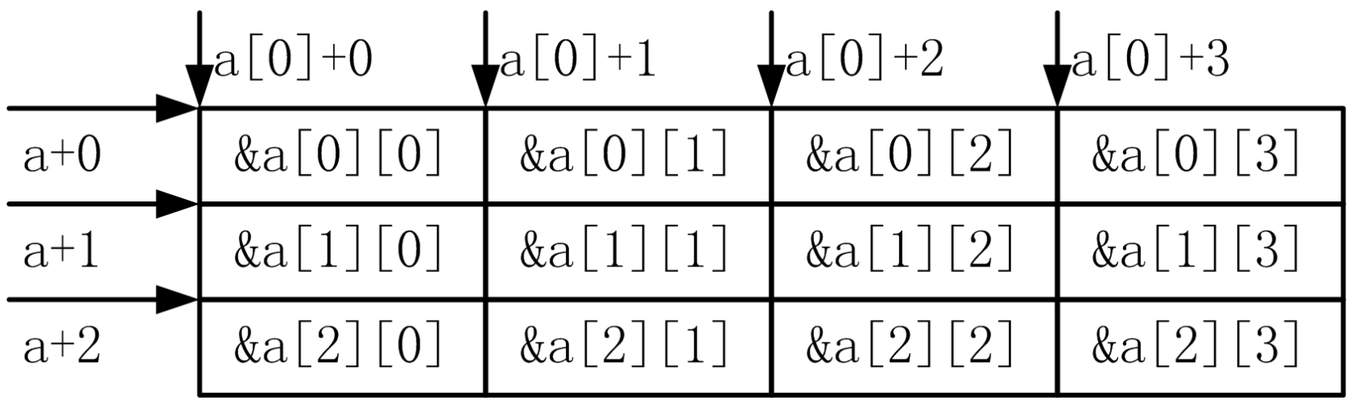
图7.12 指向二维数组的指针



（b）

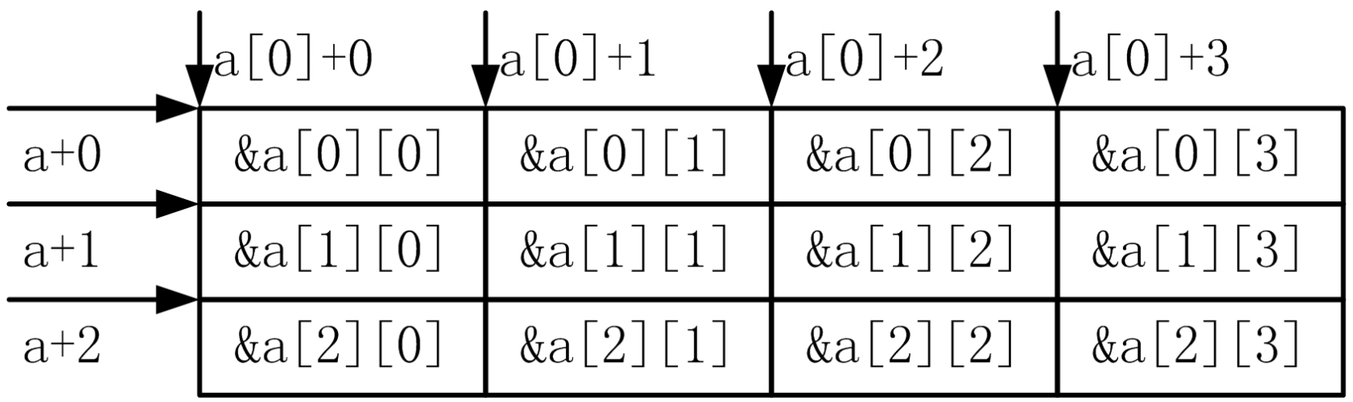
7.3.2 指向多维数组元素的指针

- a[0]、a[1]、a[2]既然是一维数组，因此a[0]既是这个“一维数组a[0]”的数组名，又是它的首地址，而“一维数组a[0]”第0个元素是a[0][0]，则a[0]与&a[0][0]等价；同理，a[1]与&a[1][0]等价，a[i]与&a[i][0]等价。



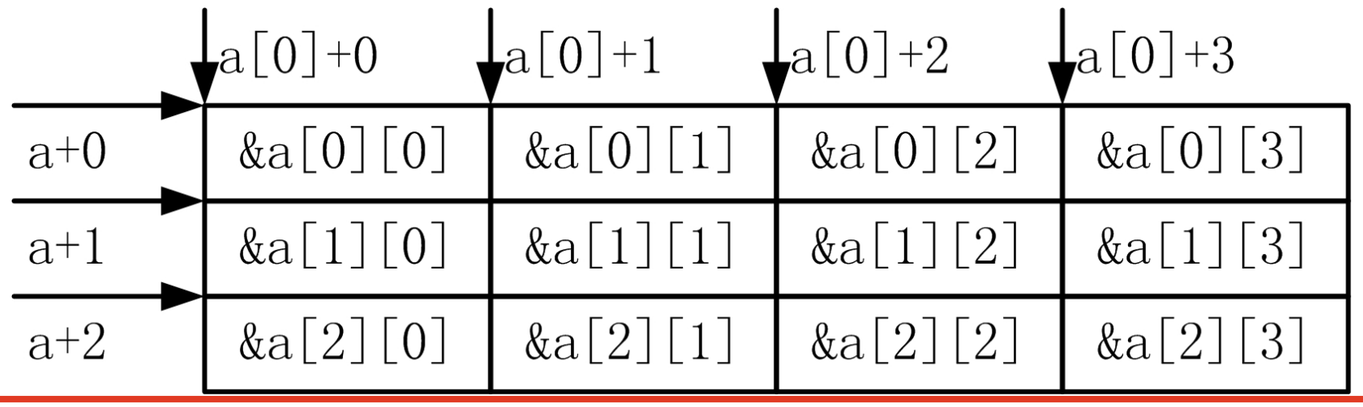
7.3.2 指向多维数组元素的指针

- &a[i]是第i行的地址，a[i]是第i行的首元素的地址，两者的值相同，但含义不一样。&a[i]指向行，a[i]指向第i行的首元素（即指向第0列）。&a[i]+1是下一行地址，而a[i]+1是第i行下一列（第1列）元素的地址。



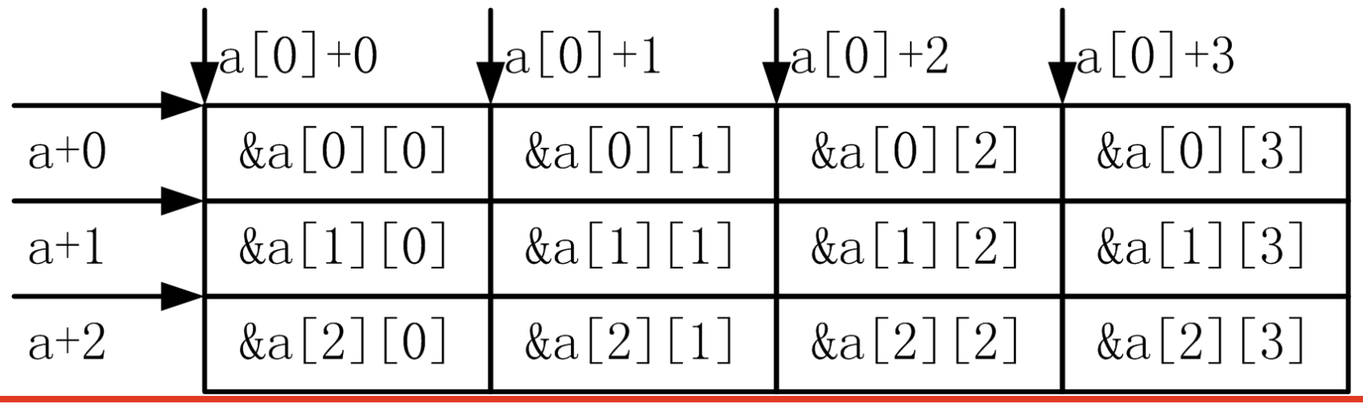
7.3.2 指向多维数组元素的指针

- 由此可知：a[0]和a+0的地址值相同，a[i]和a+i的地址值相同。前面讨论一维数组时，给出过结论：a[i]与*(a+i)等价，a+i是a[i]的地址，那这里的分析是否与此矛盾呢？其实不然，在一维数组的情形下，a+i是地址，a[i]和*(a+i)是元素；而在二维数组的情形下，a+i是地址，a[i]和*(a+i)还是地址（因为a[i]是一个数组而非元素），a[i][j]才是元素。



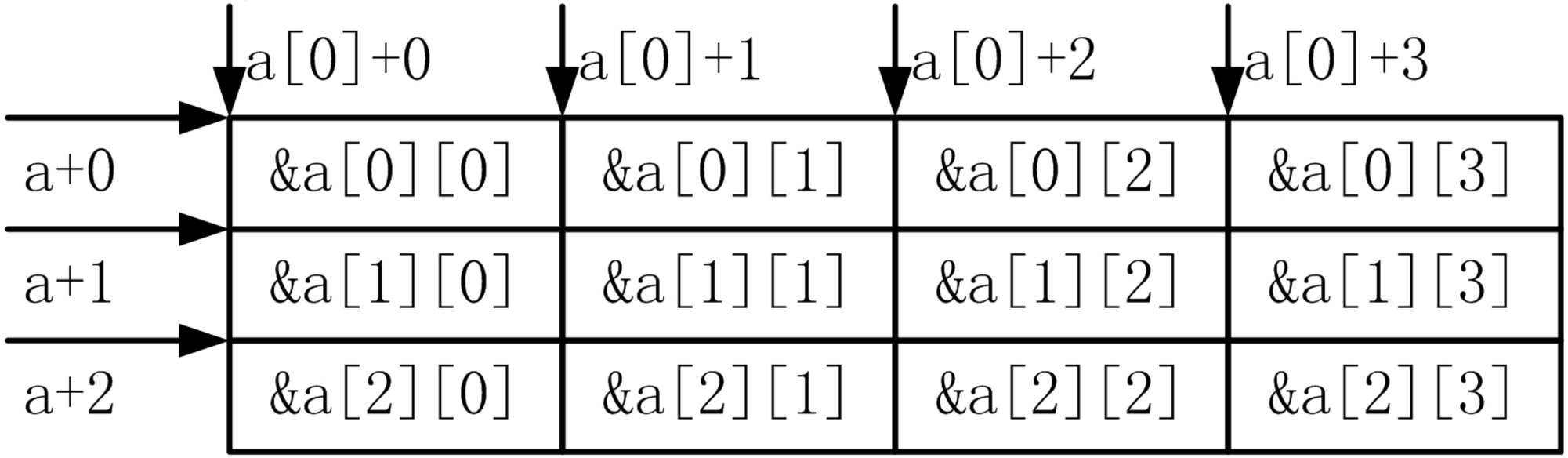
7.3.2 指向多维数组元素的指针

- 前已述及，假定有一维数组定义int B[4]，B+0是B[0]元素的地址，B+j是B[j]元素的地址。就此推理，a[0]+0（即a[0]）是a[0][0]的地址，a[0]+j是a[0][j]的地址，a[i]+j是a[i][j]的地址。由于B[0]和*(B+0)等价，B[j]和*(B+j)等价。因此a[0]+1与*(a+0)+1等价，都是&a[0][1]；a[1]+1与*(a+1)+1等价，都是&a[1][1]；a[i]+j与*(a+i)+j等价，都是&a[i][j]。



7.3.2 指向多维数组元素的指针

图7.13 二维数组地址含义



如图所示。请注意，不要将*(a+1)+1与*(a+1＋1)的写法混淆，后者是*(a+2)，相当于a[2]。

7.3.2 指向多维数组元素的指针

表7-3 二维数组的地址形式

地址形式	含义	等价地址
a	既代表二维数组对象，又是第0行首地址，即指向一维数组a[0]	&a[0][0]
a[0]、*(a+0)、*a a[i]、*(a+i)	既代表第0行（一维数组），又是第0行第0列元素的地址 既代表第i行（一维数组），又是第i行第0列元素的地址	&a[0][0] &a[i][0]
a+i、&a[i]	第i行首地址	&a[i][0]
a[i]+j、*(a+i)+j	第i行第j列元素的地址	&a[i][j]

从上述分析中可以看出，当数组是多维时，元素地址有多种等价的形式。表列出了二维数组的地址形式及其含义。

7.3.2 指向多维数组元素的指针

- 请记住，a[i]和*(a+i)是等价的，&a[i]和a+i是等价的，a[i]和*(a+i)不一定是元素，这个结论可以推广到N维数组的情形。

7.3.2 指向多维数组元素的指针

【例7.9】

输出二维数组各种形式地址值。

7.3.2 指向多维数组元素的指针

例7.9

```
#include <stdio.h>
int main()
{
   int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12} ,i=1,j=2;
   printf("a=%x\t*a=%x\n",a,*a);
   printf("a+0=%x\ta+1=%x\ta+2=%x\n",a+0,a+1,a+2);
   printf("&a[0]=%x\t&a[1]=%x\t&a[2]=%x\n",&a[0],&a[1],&a
[2]);
   printf("a[0]=%x\ta[1]=%x\ta[2]=%x\n",a[0],a[1],a[2]);
   printf("*(a+0)=%x\t*(a+1)=%x\t*(a+2)=%x\n",*(a+0),*(a+1),
*(a+2));
   printf("&a[0][0]=%x\t&a[1][0]=%x\t&a[2][0]=%x\n",&a[0]
[0],&a[1][0],&a[2][0]);
   printf("&a[i][0]=%x\t&a[i]=%x\t&a[i]+1=%x\n",&a[i][0],&a
[i],&a[i]+1);
```

7.3.2 指向多维数组元素的指针

例7.9

```
 12    printf("&a[i][0]=%x\ta[i]=%x\ta[i]+1=%x\n",&a[i][0],a[i],
a[i]+1);
 13    printf("&a[i][j]=%x\ta[i]+j=%x\t*(a+i)+j=%x\n",&a[i][j],
a[i]+j,*(a+i)+j);
 14    return 0;
 15 }
```

7.3.2 指向多维数组元素的指针

- 程序运行结果如下：

```
a=12ff50                *a=12ff50
a+0=12ff50              a+1=12ff60              a+2=12ff70
&a[0]=12ff50            &a[1]=12ff60            &a[2]=12ff70
a[0]=12ff50             a[1]=12ff60             a[2]=12ff70
*(a+0)=12ff50           *(a+1)=12ff60           *(a+2)=12ff70
&a[0][0]=12ff50         &a[1][0]=12ff60         &a[2][0]=12ff70
&a[i][0]=12ff60         &a[i]=12ff60            &a[i]+1=12ff70
&a[i][0]=12ff60         a[i]=12ff60             a[i]+1=12ff64
&a[i][j]=12ff68         a[i]+j=12ff68           *(a+i)+j=12ff68
```

7.3.2 指向多维数组元素的指针

- 2. 指向多维数组元素的指针变量
- 定义指向多维数组元素的指针变量时，指向类型应该与数组元素类型一致，例如：

```
int a[10][10], *p1=&a[0][0]; //指向二维数组元素的指针
double f[3][4][5], *p2=&f[0][0][0]; //指向三维数组元素的指针
```

7.3.2 指向多维数组元素的指针

- 3．通过指针访问多维数组元素
- 假设有

```
int a[N][M] ,*p=&a[0][0];
```

- 访问一个二维数组元素a[i][j]，可以用：
- （1）数组下标法：a[i][j]；
- （2）指针下标法：p[i*M+j]；
- （3）地址引用法：*(*(a+i)+j)或*(a[i]+j)；
- （4）指针引用法：*(p+i*M+j)。

7.3.2 指向多维数组元素的指针

- 由于指针变量p指向类型为int，说明p指向的一定是数组元素。当通过p来访问二维数组或多维数组时，本质上是将多维数组按一维数组来处理，因而它的访问方法与一维数组类似，只不过需要计算a[i][j]元素在数组中的相对位置，其公式为：

$$i * M + j$$

- 其中M为二维数组第2维数组的长度。

7.3.2 指向多维数组元素的指针

【例7.10】

通过指针变量遍历二维数组元素。

7.3.2 指向多维数组元素的指针

例7.10

```
1 #include <stdio.h>
2 int main()
3 {
4   int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12},*p;
5   for (p=a[0]; p<a[0]+12; p++) {
6     printf("%2d ",*p);
7     if ((p-a[0])%4==3) printf("\n"); //每行输出4个元素后换行
8   }
9   return 0;
10 }
```

循环初始p=a[0]，p指向a[0][0]元素，p++指向后面连续的元素，a[0]+12为最后一个元素a[2][3]后面元素的地址，p-a[0]为p当前指向的元素与a[0][0]之间的元素个数。

7.3.2 指向多维数组元素的指针

例7.10

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

程序运行屏幕

```
1  2  3  4
5  6  7  8
9 10 11 12
```

7.3.2 指向多维数组元素的指针

- 由于多维数组的地址形式有多种，因此指针变量初始指向存在多种写法，使得指针运算后的指向是比较复杂的。例如：

```
int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12},*p;
p=&a[2][2]; //正确，p指向a[2][2]
p=a[0]+5; //正确，p指向a[1][1]
p=*(a+2)+4; //正确，p指向a[2][4]
p=a; //错误，指向类型不相同 a为指向第0行
p=&a[1]; //错误，指向类型不相同，&a[1]为指向第1行
p=a+5; //错误，指向类型不相同，且a+5指向第5行（无效地址）
```


程序设计