

计算机导论与程序设计 [CS006001-60]

段江涛

机电工程学院



2019 年 10 月

定义和引用二维数组: `int a[10][20];`
○○○

引用数组: `int i=0,j=0; a[i][j]`
○○

二维数组的初始化: `int a[2][3]={ {1,2,3}, {4,5,6}, {7,8,9} }`
○○○○○

用字符数组表示字符串 `char s[81]`
○○○○○

lecture-11 主要内容

二维数组和字符数组

- 1 定义和引用二维数组: `int a[10][20];`
- 2 引用数组: `int i=0,j=0; a[i][j]`
- 3 二维数组的初始化: `int a[2][3]={ {1,2,3}, {4,5,6}, {7,8,9} }`
- 4 用字符数组表示字符串 `char s[81]`

二维数组应用场景

Example

有 3 个小分队,每队有 6 名队员,要把这些队员的工资用数组保存起来以备查。

	队员 1	队员 2	队员 3	队员 4	队员 5	队员 6
第 1 分队	1000	2000	1500	2400	3000	5000
第 2 分队	3000	4000	2500	2300	2500	4000
第 3 分队	4000	5000	1200	2300	3200	5500

`float pay[3][6];` // 定义3行6列的二维数组,称为3×6矩阵(matrix)。

<code>pay[0][0]</code>	<code>pay[0][1]</code>	<code>pay[0][2]</code>	<code>pay[0][3]</code>	<code>pay[0][4]</code>	<code>pay[0][5]</code>
<code>pay[1][0]</code>	<code>pay[1][1]</code>	<code>pay[1][2]</code>	<code>pay[1][3]</code>	<code>pay[1][4]</code>	<code>pay[1][5]</code>
<code>pay[2][0]</code>	<code>pay[2][1]</code>	<code>pay[2][2]</code>	<code>pay[2][3]</code>	<code>pay[2][4]</code>	<code>pay[2][5]</code>

定义和引用二维数组: int a[10][20];

○○○

引用数组: int i=0,j=0; a[i][j]

○○

二维数组的初始化: int a[2][3]={ {1,2,3}, {4,5,6}, {7,8,9} }

○○○○○

用字符数组表示字符串 char s[81]

○○○○○

定义二维数组: int a[10][20];

定义二维数组:

元素类型 数组名 [行数][列数]

int a[5][10]; // 定义5行10列的整型二维数组

float pay[3][6]; // 定义3行6列的单精度浮点型二维数组

double pay1[3][6]; // 定义3行6列的双精度浮点型二维数组

char c[2][20]; // 定义2行20列的字符型二维数组

// 行数列数定义为常量

#define M 5 // 行数

#define N 10 // 列数

int a[M][N]; // 定义M行N列的整型二维数组

二维数组的存储

```
float a[3][4]; // 定义3行4列的单精度浮点型二维数组
```

Notes

用矩阵形式(如3行4列形式)表示二维数组,是逻辑上的概念,能形象地表示出行列关系。而在内存中,各元素是连续存放的,不是二维的,是线性的。

数组元素的下标从0开始, `int a[5][10]`; 50个整型元素,先行后列存储数组元素,第1行第1列元素是 `a[0][0]`,第5行第10列元素是 `a[4][9]`。

<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>	⇐ 第0行
<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>	⇐ 第1行
<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>	⇐ 第2行

1000	<code>a[0][0]</code>
1004	<code>a[0][1]</code>
1008	<code>a[0][2]</code>
1012	<code>a[0][3]</code>
1016	<code>a[1][0]</code>
1020	<code>a[1][1]</code>
1024	<code>a[1][2]</code>
1028	<code>a[1][3]</code>
1032	<code>a[2][0]</code>
1036	<code>a[2][1]</code>
1040	<code>a[2][2]</code>
1044	<code>a[2][3]</code>

定义和引用二维数组: int a[10][20];
○○○

引用数组: int i=0,j=0; a[i][j]
●○

二维数组的初始化: int a[2][3]={ {1,2,3}, {4,5,6}, {7,8,9} }
○○○○○

用字符数组表示字符串 char s[81]
○○○○○

引用数组: int i=0,j=0; a[i][j]

```
#define M 5 // 行数
#define N 10 // 列数

float a[M][N]; // 定义M行N列的单精度浮点型二维数组
int i,j;

a[0][0]=10; // 给第0行第0列元素赋值
a[M-1][N-1]=20; // 给第M-1行第N-1列元素赋值

// 数组元素赋值                                // 输出数组元素
for(i=0;i<M;i++) //遍历第i行                    for(i=0;i<M;i++)
    for(j=0;j<N;j++) //遍历第j列                for(j=0;j<N;j++)
        a[i][j]=i*j; //给第i行第j列元素赋值    {
                                                    printf("%d\t",a[i][j]);
                                                    if(j==N-1) printf("\n");
                                                    }

// 键盘输入数组元素
for(i=0;i<M;i++)
    for(j=0;j<N;j++)
        scanf("%f", &a[i][j]);
```

输出每个小分队的平均工资

```
#define M 100 // 估计最大的行数
#define N 100 // 估计最大的列数

float pay[M][N]; //按照最大值定义数组的大小
int m,n,i,j; //m,n表示实际pay数组行数和列数
float sum; //每个小分队所有人员的工资总和

scanf("%d%d",&m,&n); // 输入小分队数和队员数, 就是pay数组的实际行数和列数

for(i=0;i<m;i++) // 依次输入每个队员的工资
    for(j=0;j<n;j++) scanf("%f",&pay[i][j]); // 注意取地址符'&'

for(i=0;i<m;i++)
{
    sum=0; // 内层循环执行前, 必须置零。易忘记!!!
    for(j=0;j<n;j++)
        sum += pay[i][j]; // 累加第i小分队所有人员的工资
    printf("第%d小分队平均工资=%f\n",i+1,sum/n);
}
```

pay[0][0]	pay[0][1]	pay[0][2]	pay[0][3]	pay[0][4]	pay[0][5]
pay[1][0]	pay[1][1]	pay[1][2]	pay[1][3]	pay[1][4]	pay[1][5]
pay[2][0]	pay[2][1]	pay[2][2]	pay[2][3]	pay[2][4]	pay[2][5]

二维数组的初始化: int a[2][3]={ {1,2,3}, {4,5,6}, {7,8,9} }

```
#define M 3 // 行数
#define N 3 // 列数

int a[M][N]={ {1,2,3}, {4,5,6}, {7,8,9} }; // 定义二维数组, 并初始化
int b[M][N]={ {1,2}, {4,5}, {7,8} }; // 定义二维数组, 并初始化
int c[M][N]={ {1,2}, {}, {7,8} }; // 定义二维数组, 并初始化

// 给未初始化的元素赋值
b[0][N-1]=3; b[1][N-1]=6; b[2][N-1]=9;
c[0][N-1]=3;
c[1][0]=4; c[1][1]=5; c[1][N-1]=6;
c[2][N-1]=9;
```


例 6.4 将二维数组行列互换, 存到另一个二维数组中。

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \Rightarrow b = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
#define M 2
#define N 3
int a[M][N]={ {1,2,3}, {4,5,6} };
int b[N][M],i,j;
```

```
for(i=0;i<M;i++)
{
    for(j=0;j<N;j++)
    {
        printf("%5d",a[i][j]);
        b[j][i]=a[i][j];
    }
    printf("\n"); // 换行
}
```

```
for(i=0;i<M;i++)
{
    for(j=0;j<N;j++)
    {
        printf("%5d",b[i][j]);
    }
    printf("\n"); // 换行
}
```

例 6.4 将二维数组行列互换, 存到另一个二维数组中。

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \Rightarrow b = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
#define M 2
#define N 3
int a[M][N]={ {1,2,3}, {4,5,6} };
int b[N][M],i,j;
```

```
for(i=0;i<M;i++)
{
    for(j=0;j<N;j++)
    {
        printf("%5d",a[i][j]);
        b[j][i]=a[i][j];
    }
    printf("\n"); // 换行
}
```

```
for(i=0;i<M;i++)
{
    for(j=0;j<N;j++)
    {
        printf("%5d",b[i][j]);
    }
    printf("\n"); // 换行
}
```

例 6.4 将二维数组行列互换, 存到另一个二维数组中。

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \Rightarrow b = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
#define M 2
#define N 3
int a[M][N]={ {1,2,3}, {4,5,6} };
int b[N][M],i,j;
```

```
for(i=0;i<M;i++)
{
    for(j=0;j<N;j++)
    {
        printf("%5d",a[i][j]);
        b[j][i]=a[i][j];
    }
    printf("\n"); // 换行
}
```

```
for(i=0;i<M;i++)
{
    for(j=0;j<N;j++)
    {
        printf("%5d",b[i][j]);
    }
    printf("\n"); // 换行
}
```

例 6.4 将二维数组行列互换, 存到另一个二维数组中。

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \Rightarrow b = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
#define M 2
#define N 3
int a[M][N]={ {1,2,3}, {4,5,6} };
int b[N][M],i,j;
```

```
for(i=0;i<M;i++)
{
    for(j=0;j<N;j++)
    {
        printf("%5d",a[i][j]);
        b[j][i]=a[i][j];
    }
    printf("\n"); // 换行
}
```

```
for(i=0;i<M;i++)
{
    for(j=0;j<N;j++)
    {
        printf("%5d",b[i][j]);
    }
    printf("\n"); // 换行
}
```

例 6.5 求二维数组中的最大元素及其行号和列号。

```
#define M 2
#define N 3
int a[M][N]={ {1,2,3}, {4,5,6} }, i, j, max, row=0, col=0;
max=a[0][0]; // //先认为a[0][0]最大
for(i=0; i<M; i++)
{
    for(j=0; j<N; j++)
    {
        if(a[i][j]>max)
        {
            max=a[i][j]; row=i; col=j;
        }
    }
}
printf("max=%d, row=%d, col=%d\n", max, row, col);
```

例: 方阵 int a[10][10] 左右对角线

例: 使方阵 (行数 = 列数) 左对角线元素置 1, 其它元素为 0.

```
#define M 10                                     // 输出
int a[M][M], i, j;                               for(i=0; i<M; i++) // 行
for(i=0; i<M; i++)                               {
{
    for(j=0; j<M; j++) // 列
        printf("%5d", a[i][j]);
    printf("\n"); // 换行
    //右对角线, if(i+j==M)
    if(i==j) a[i][j]=1;
    else a[i][j]=0;
}
}
```

例: 方阵 int a[10][10] 上下三角阵

例: 使方阵 (行数 = 列数) 下三角阵元素置 1, 其它元素为 0.

```
#define M 10

int a[M][M], i, j;

for(i=0; i<M; i++) // 行
    for(j=0; j<M; j++) // 列
    {
        // 如果是上三角阵, if(j>=i)
        if(j<=i) a[i][j]=1;
        else a[i][j]=0;
    }

printf("下三角阵\n");
for(i=0; i<M; i++) // 行
{
    for(j=0; j<=i; j++) // 列
        printf("%5d", a[i][j]);
    printf("\n"); // 换行
}
```

```
printf("上三角阵\n");
for(i=0; i<M; i++) // 行
{
    for(j=0; j<M; j++) // 列
        if(j>=i) printf("%5d", a[i][j]);
        else printf("%5c", ' ');
    printf("\n"); // 换行
}
```

```
下三角阵
1
1 1
1 1 1
1 1 1 1
1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1

上三角阵
1 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0
  1 0 0 0 0 0 0 0 0 0
    1 0 0 0 0 0 0 0 0 0
      1 0 0 0 0 0 0 0 0 0
        1 0 0 0 0 0 0 0 0 0
          1 0 0 0 0 0 0 0 0 0
            1 0 0 0 0 0 0 0 0 0
              1 0 0 0 0 0 0 0 0 0
                1 0 0 0 0 0 0 0 0 0
```

定义和引用二维数组: int a[10][20];
○○○

引用数组: int i=0,j=0; a[i][j]
○○

二维数组的初始化: int a[2][3]={ {1,2,3}, {4,5,6}, {7,8,9} }
○○○○○

用字符数组表示字符串 char s[81]
●○○○○

用字符数组表示字符串 char s[81]

字符数组是用来存放字符数据的数组是字符数组。在字符数组中的一个元素内存放一个字符。

两种方式初始化字符数组

```
// s[0]='a', s[1]='b', s[2]='c', s[3]='d', s[4]以后的字符未赋值
```

```
char s[81]={ 'a', 'b', 'c', 'd' }; // 4个有效字符
```

```
// str[0]='a', str[1]='b', str[2]='c', str[3]='d', str[4]='\0'
```

```
// 自动追加'\0', 表示字符串结束)
```

```
char str[]="abcd"; // 5个有效字符
```


使用 printf 函数输出字符串

```
// s[0]='a', s='b', s='c', s[3]='d', s[4]以后的字符未赋值
```

```
char s[81]={'a','b','c','d'}; // 4个有效字符
```

```
for(i=0;i<4;i++)
```

```
    printf("%c\t",s[i]); // a b c d
```

```
// str[0]='a', str[1]='b', str[2]='c', str[3]='d', str[4]='\0'
```

```
// 自动追加'\0', 表示字符串结束)
```

```
char str[]="abcd"; // 5个有效字符
```

```
for(i=0;a[i]!='\0';i++)
```

```
    printf("%c\t",str[i]); // a b c d
```

```
// 或使用格式描述符%s, 输出以'\0'结尾的字符串
```

```
printf("%s\n",str); // abcd, 输出'\0'以前的字符
```

```
printf("%s\n",s); // 错误, 由于s不是以'\0'结尾的字符串
```

```
s[4]='\0'; // 使字符数组以'\0'结尾, 就可以使用上一句正常输出了
```

使用 scanf 函数输入字符串

```
char a[81],b[81],c[81];
```

```
// 遇空格或回车结束, 自动追加字符串结束字符'\0'
```

```
scanf("%s",a); // 注意字符数组前没有取地址符号'&'
```

```
// 例如输入: abc回车, 则a[0]='a',a[1]='b',a[2]='c',a[3]='\0'
```

```
printf("%s\n",a); // abc
```

```
// 以空格隔开, 输入3个字符串。自动追加字符串结束字符'\0'
```

```
scanf("%s%s%s",a,b,c);
```

```
// 例如输入: How are You回车
```

```
printf("%s,%s,%s\n",a,b,c); // How,are,You
```

推荐使用 gets 函数和 puts 函数输入输出字符串

```
char a[81];
```

```
// 可以接收带空格的字符串，遇回车结束，自动追加字符串结束字符'\0'
```

```
gets(a);
```

```
// 例如输入: ab cd回车
```

```
// 遇字符串结束字符'\0'，换行退出
```

```
puts(a); // ab cd
```

欢迎批评指正！