- 基本要求：
- 　　１）熟悉二维及多维数组的定义、初始化与引用；
- 　　２）熟悉字符串数组与字符串操作；
- 　　３）熟悉“函数与数组”的相关操作。
- 学习重点：
- 　　１）二维及多维数组的定义、初始化与引用；
- 　　２）字符串数组与字符串操作；
- 　　３）“函数与数组”的相关操作；
-

7.1 二维数组
7.1.1 二维数组的定义

- 当数组元素具有两个下标时，该数组称为二维数组。二维数组可以看作具有行和列的平面数据结构，如矩阵。二维数组的一般格式为：

  数据类型 数组名[常量表达式1][常量表达式2]

- ① 数据类型表示数组中数据的存储类型，“常量表达式1”表示数组的行数，“常量表达式2”表示数组的列数。
- ② 元素个数=行数×列数。
- ③ 常量表达式是常量或符号常量，不能为变量。

7.1.1 二维数组的定义

- 我们可以把二维数组看作是一种特殊的一维数组：它的元素又是一个一维数组。例如：float a[3][4]；可以把a看作是一个一维数组，它有3个元素：a[0]、a[1]、a[2]，每个元素又是一个包含4个元素的一维数组。可以把a[0]、a[1]、a[2]看作是一维数组的名字。
- C语言中，由于内存是一维的，二维数组中元素排列的顺序是：按行存放，即在内存中先顺序存放第一行的元素，再存放第二行的元素。
- 数组名a代表a数组的首地址。通常形象地把第一个下标称为行下标，第二个下标称为列下标。
- 数组a[3][4]在内存中的存放顺序为：

  a[0][0] a[0][1] a[0][2] a[0][3]
  a[1][0] a[1][1] a[1][2] a[1][3]
  a[2][0] a[2][1] a[2][2] a[2][3]

7.1.2 二维数组的初始化

- 二维数组初始化的格式为：
  数据类型 数组名[整常量表达式][整常量表达式]={初始化数据};
  在{ }中给出各数组元素的初值，各初值之间用逗号分开，{ }中的初值依次赋给各数组元素。
- 1．按行赋初值
  例如：
  int a[2][3]={{1,2,3},{4,5,6}};
  这种方法非常直观，把第一个花括号内的数据赋给第一行的元素，第二个花括号内的数据赋给第二行的元素，…。
- 2．按元素排列顺序全部赋初值
  例如：
  int a[2][3]={1,2,3,4,5,6};
  这种方法把所有数据写在一个花括号内。如果数据较多，容易遗漏数据，而且不易检查。

7.1.2 二维数组的初始化

- 3．部分元素赋初值
- ① 的每个花括号表示一行，对各行中的某一元素赋初值。
  例如：int a[2][3]={{1,2},{4}};
- ② 只赋初值第一行。
  例如：int a[2][3]={1,2,4};
- ③ 对每行第一个元素赋初值。
  例如：int a[3][4]={{1},{2},{3}};
- ④ 第二行不赋初值。
  例如：int a[3][4]={{1,2},{ },{9}};
- ⑤ 部分元素赋初值而省略第一维的长度，但分行赋初值；全部元素都赋初值，则定义数组时第一维长度可以省略，但第二维的长度必须指定。
  例如：int a[][3]={{1},{4,5}};等价于int a[2][3]={{1},{4,5}};。
  int a[][3]={1,2,3,4,5,6};等价于int a[2][3]={1,2,3,4,5,6};。

7.1.2 二维数组的初始化

- 例如：
- for (i=0;i<=100;i++)
- for(j=0;j<=200;j++)
- a[i][j]=i+j;
- for (i=0;i<=100;i++)
- for(j=0;j<=200;j++)
- scanf(“%d”,&a[i][j]);

7.1.3 二维数组元素的引用

- 二维数组元素的引用格式为：
  数组名[下标][下标]
- ① 下标可以是整型表达式。
  例如：
  ```
  int a[2][3],i=1,j=2,k=0;
  a[i][k]=a[i-1][j]+a[1][j];
  ```
  表示a[0][2]的值与a[1][2]的值求和并赋给a[1][0]。
- ② 数组元素可以出现在表达式中，也可以被赋值。
  例如：
  ```
  b [1][2]= a[2][3]/2
  ```
- ③ 引用数组元素时，要注意下标值不能超过数组的范围。
  例如：
  ```
  int a[3][4];
  a[3][4]=3;
  ```
  定义a为3×4的数组，行下标值最大为2，列下标值最大为3。用a[3][4]超过了数组的范围。定义数组时的a[3][4] 用来定义数组a的维数和各维的长度，而引用元素时的a[3][4] 中的3和4是下标值，a[3][4]代表某一个元素，一定要注意它们之间的区别。
- ④ 数组不能整体使用，只能逐个引用数组元素。

7.1.3 二维数组元素的引用

例7-1 用二维数组保存三个班的英语成绩（每个班20人），并求每个班的平均成绩。

```
#include <stdio.h>
main( )
{    float  score[3][20], sum[3]={0,0,0}, aver[3];
      int i,j;
      for (i=0;i<3;i++)
      {     printf ("请输入第%d个班20个人的成绩：\n", i+1);
            for (j=0;j<20;j++)
                  scanf("%f",&score[i][j]);
      }
      for (i=0;i<3;i++)
      {     for (j=0;j<20;j++)
                  sum[i]=sum[i]+score[i][j];
            aver[i]=sum[i]/20 ;
            printf("第%d个班的英语平均成绩为：%f\n",i+1,aver[i]);
       }
}
```

7.1.3 二维数组元素的引用

【例7-2】 编写程序生成4×4矩阵A，并将行和列元素互换后生成矩阵B，并将矩阵A和B输出显示。

分析：A、B矩阵的行和列元素互换，可以得出它们的对应关系：

b[i][j] = a[j][i]（i=0，1，2，3；j=0，1，2，3）

```
#define N 4
main()
{
    int i,j,k;
    int a[N][N],b[N][N];
    for(i=0;i<N;i++)                                        /* 输入A矩阵元素 */
        for (j=0;j<N;j++)
            scanf("%d",&a[i][j]);
    printf("A array:\n");
    for(i=0;i<N;i++)                                    /* 输出A矩阵 */
        {for(j=0;j<N;j++)
            printf("%-4d", a[i][j]);
            printf("\n");                                /* 每输出一行,立即换行 */
        }
    for(i=0;i<N;i++)                                    /* 产生B矩阵元素 */
        for(j=0;j<N;j++)
          b[i][j]=a[j][i];
    printf("B array:\n");
    for(i=0;i<N;i++)                                    /* 输出B矩阵 */
        {for(j=0;j<N;j++)
            printf("%-4d", b[i][j]);
            printf("\n");                                    /* 每输出一行,立即换行 */
        }
}
```

7.2 字符数组

- 数据元素是字符型数据的数组称为字符数组。字符数组中的一个元素存放一个字符，数组名表示数组中第一个字符的地址。

7.2.1 字符数组的定义

- 字符数组的定义方法与数值数组的定义类似，只要把数据类型改为char即可。

  例如：　　char c[10];

  　　c[0]='I';c[1]=' ';c[2]='a';c[3]='m';c[4]=' ';c[5]='a';c[6]=' ';c[7]='b';c[8]='o';c[9]='y';
- 因为字符型与整型是互相通用的，上面的定义也可改为int c[10];