数组

为什么需要数组

为了解决大量同类型数据的存储和使用问题；
为了模拟现实世界。

数组的分类

一维数组

怎样定义一维数组

为 n 个变量连续分配存储空间；
所有的变量数据类型必须相同；
所有变量所占的字节大小必须相等。

例子：

```
Int[5];
//一维数组名不代表数组中的所有元素
//一维数组名代表数组第一个元素的地址
```

有关一维数组的操作：

初始化

①、完全初始化：

```
int a[5] = {1, 2, 3, 4, 5};
```

②、不完全初始化(未被初始化的元素自动为零)：

```
int a[5] = {1, 2, 3};
```

③、不初始化(所有元素是垃圾值)：

```
Int a[5];
```

④、清零(里面的值都初始化为 0)：

```
Int a[5] = {0};
```

错误写法：

```
Int a[5];//定义的时候方括号内的值是个数
a[5] = {1, 2, 3, 4, 5};//错误 //定义完之后，方括号内的值是下标
```

只有在定义数组的同时才可以整体赋值,其他情况下整体赋值都是错误的。

```
Int a[5] = {1, 2, 3, 4, 5};
a[5] = 100;//error,因为没有 a[5]这个元素，最大只有 a[4]。

Int a[5] = {1, 2, 3, 4, 5};
Int b[5];
```

如果要把 a 数组中的值全部赋给 b 数组：

错误的写法：

```
b = a; //error
```

正确的写法：

```
for(i=0; i<5; i++)
    b[i] = a[i];
```

赋值
排序
求最大/最小值
倒置
查找

二维数组

int a[3][4];

总共有 12 个元素，可以当做 3 行 4 列看待，这 12 个元素的名字依次是：

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]
a[i][j]表示第 i+1 行，j+1 列的元素。
int[] a[m][n];最大位置的元素(右下角)只能是 a[m-1][n-1];

初始化：

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

操作：

输出二维数组的内容：

```c
#include <stdio.h>
int main(void) {

    int a[3][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
    };
    int i, j;
    for (i=0; i<3; i++) {
        for (j=0; j<4; j++)
        printf("%-5d  ", a[i][j]);//-5d表示对齐，-表示左对齐，5表示这个值占5个光标的位置。
        printf("\n");
    }
    return 0;
}
```

//通过两个 for 循环输出。

对二维数组排序
求每一行的最大值
判断矩阵是否对称
矩阵的相乘

多维数组

是否存在多维数组？

不存在；
因为内存是线性一维的；
n 维数组可以当作每个元素是 n-1 维数组的一维数组。
比如：

int a[3][4];

该数组是含有 3 个元素的一维数组；

只不过每个元素都可以分成 4 个小元素。

int a[3][4][5];

该数组是含有 3 个元素的一维数组；

只不过每个元素都是 4 行 5 列的二维数组。

int a[5];//如果 int 占 4 个字节，则本数组包含 20 个字节，每 4 个字节被当作一个 int 变量来使用。