数组

概念

> 数组是一组**相同类型元素**的集合
>
> 可以存放一个或多个数据，但是不能是零个
>
> 分为**一维数组**和**多维数组**

一维数组

一维数组的创建

```
//创建一个数组，存放班级中50个人的数学成绩
int math[50];
```

数组的初始化

```
//完全初始化，完完全全塞满整个数组
int array[5] = { 1,2,3,4,5 };
```

```
//数组的不完全初始化，剩下的默认初始化为0
int a[5] = { 1 };
```

数组的下标 & 使用

> 可以将下标理解成**门牌号**
>
> 数组通过下标访问具体元素，下标也叫**索引**

整形数组

```
int main() {
    int a[5] = { 1,2,3,4,5 };
    //           0 1 2 3 4
    printf("%d", a[0]);
    printf("%d", a[1]);
    printf("%d", a[2]);
    printf("%d", a[3]);
    printf("%d", a[4]);
    a[2] = 30;                    //重新为a[2]赋值
    printf(" %d",a[2]);
    return 0;
}
输出结果：
12345 30
```

```c
//通过循环输出
int main() {
    int a[5] = { 1,2,3,4,5 };
    for ( int i = 0; i < 5; i++)
    {
        printf("%d ", a[i]);
    }
    return 0;
}
```

```c
//输入数据
//整型数组 的操作是逐个元素去处理的，没办法一次性输入，也没办法一次性输出，必须使用循环
int main() {
    int a[] = { 1,2,3,4,5 };
    for (int i = 0; i < 5; i++)
    {
        scanf("%d",&a[i]);                //输入，这是一个数据元素，所以要取地址
    }
    for ( int i = 0; i < 5; i++)
    {
        printf("%d ", a[i]);
    }
    return 0;
}
```

> **整型数组 的操作是逐个元素去处理的，没办法一次性输入，也没办法一次性输出，必须使用循环**

二分查找

```c
//二分查找
#include <stdio.h>
#include <stdbool.h>
int main() {
    int n = 0;
    int arr[10] = { 1,2,3,4,5,6,7,8,9,10 };
    scanf("%d",&n);                                    //输入要查找的数
    int len = sizeof(arr) / sizeof(arr[0]);            //计算数组长度
    int left = 0,right = len - 1;                      //初始化最左边和最右边
    bool flag = false;                                 //定义一个flag，判断最后到底
找没找到
    while (left <= right) {

        //int mid = (left + right) / 2;                     //计算中间值
        // 改进计算中间值的方式，避免整数溢出
        int mid = left + (right - left) / 2;

        if (arr[mid] < n) {                            //如果数组的中间值小于要查找的
值，就说明要查找的数字在右边
            left = mid + 1;
        }
        else if (arr[mid] > n) {                       //如果数组的中间值大于要查找的
值，就说明要查找的数字在左边
            right = mid - 1;
        }
```

```
        else {
            printf("找到了！，下标为%d", mid);
            flag = true;
            break;
        }
    }
    if (!flag)                                          //如果全部查完后依旧没有找到，
flag = false 输出未找到
        printf("未找到%d",n);
    return 0;
}
```

字符数组

> 一维数组中的所有元素在内存中是**连续存放**的，随着数组下标的增长而增长

数组在内存中的关系

```
#include <stdio.h>
int main() {
    int arr[5] = { 1,2,3,4,5 };
    for (int i = 0; i < 5; i++)
    {
        printf("arr[%d] = %p\n", i, &arr[i]);
    }
    return 0;
}
```

```
输出结果：
//x86架构
arr[0] = 010FFD8C       //每次加4
arr[1] = 010FFD90
arr[2] = 010FFD94
arr[3] = 010FFD98
arr[4] = 010FFD9C

//x64架构
arr[0] = 0000009FE80FFCA8
arr[1] = 0000009FE80FFCAC
arr[2] = 0000009FE80FFCB0
arr[3] = 0000009FE80FFCB4
arr[4] = 0000009FE80FFCB8
```

> x86和x64位的差别是一个是32位一个是64位

---

sizeof计算数组元素个数

```
#include <stdio.h>
int main() {
    int a[10] = { 0 };
    printf("数组的总长度为：%zu\n",sizeof(a));                    //计算数组a的总长度，10 * 4 = 40 字节
    printf("数组的个数为：%zu\n", sizeof(a) / sizeof(a[0]));        //数组的总长度 ÷ 单个数组元素的长度
    return 0;
}
```

```
输出结果：
数组的总长度为：40
数组的个数为：10
```

二维数组

二维数组的创建

> 二维数组的每个元素是一维数组

```
int arr[2][4];          //二行四列
double data[2][8];
```

二维数组的初始化

```
//不完全初始化
#include <stdio.h>
int main() {
    int arr[3][5] = {1,2,3};         //不完全初始化，只初始化前三个元素，其他初始化为零
    return 0;
}
```

```
//完全初始化
#include <stdio.h>
int main() {
    int arr[2][3] = { 1,2,3,4,5,6 };
    return 0;
}
```

```
//初始化的时候可以省略行，但是不能省略列
int arr5[][5] = {1, 2, 3};
int arr6[][5] = {1, 2, 3, 4, 5, 6, 7};
int arr7[][5] = {{1, 2}, {3, 4}, {5, 6}};
```

二维数组的使用

> 二维数组也是有下标的，通过行号和列号来访问

```
#include <stdio.h>
int main() {
    int arr[3][4] = { {1,2,3,4},
                      {5,6,7,8},
                      {9,10,11,12} };
    //找7
    printf("%d", arr[1][2]);
    return 0;
}
```

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

    for (int i = 0; i < 3; i++)
    {
        for(int j = 0;j < 4; j++)
        {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
输出结果：
1 2 3 4
5 6 7 8
9 10 11 12
```

输入输出

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

```
}
```

二维数组在内存中关系

```
#include <stdio.h>
int main() {
    int arr[3][4];                          //因为定义的是int类型，int类型长度为4字节
    for (int i = 0; i < 3; i++){
        for (int j = 0; j < 4; j++) {
            scanf("%d",&arr[i][j]);
        }
    }
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            printf("&arr[%d][%d] = %p\n",i,j,&arr[i][j]);
        }
        //printf("\n");
    }
    return 0;
}
```

```
&arr[0][0] = 00ECFEA4
&arr[0][1] = 00ECFEA8
&arr[0][2] = 00ECFEAC
&arr[0][3] = 00ECFEB0
&arr[1][0] = 00ECFEB4
&arr[1][1] = 00ECFEB8
&arr[1][2] = 00ECFEBC
&arr[1][3] = 00ECFEC0
&arr[2][0] = 00ECFEC4
&arr[2][1] = 00ECFEC8
&arr[2][2] = 00ECFECC
&arr[2][3] = 00ECFED0
```