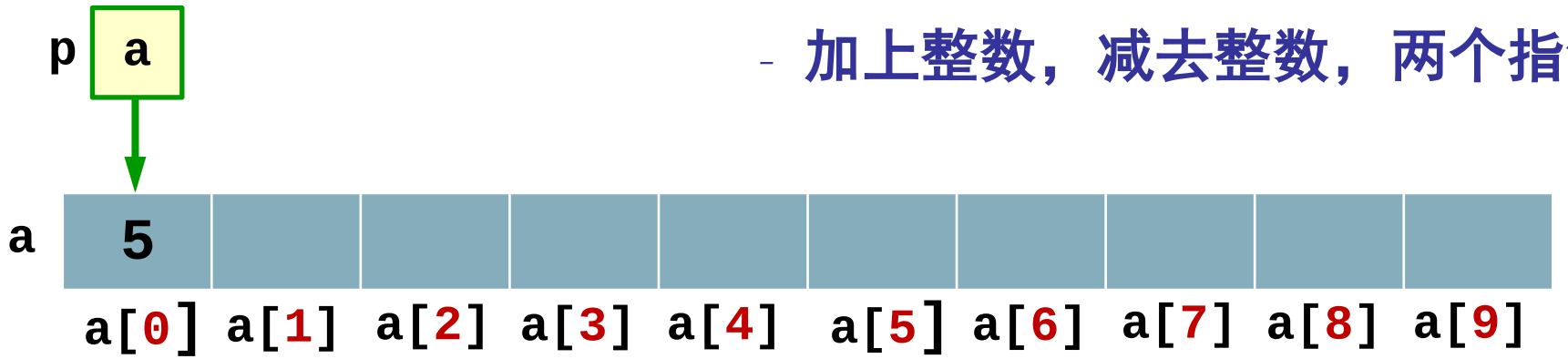
指针的算术运算

```
int a[10];
int *p = a;
int *p = &a[0];
*p = 5;
```

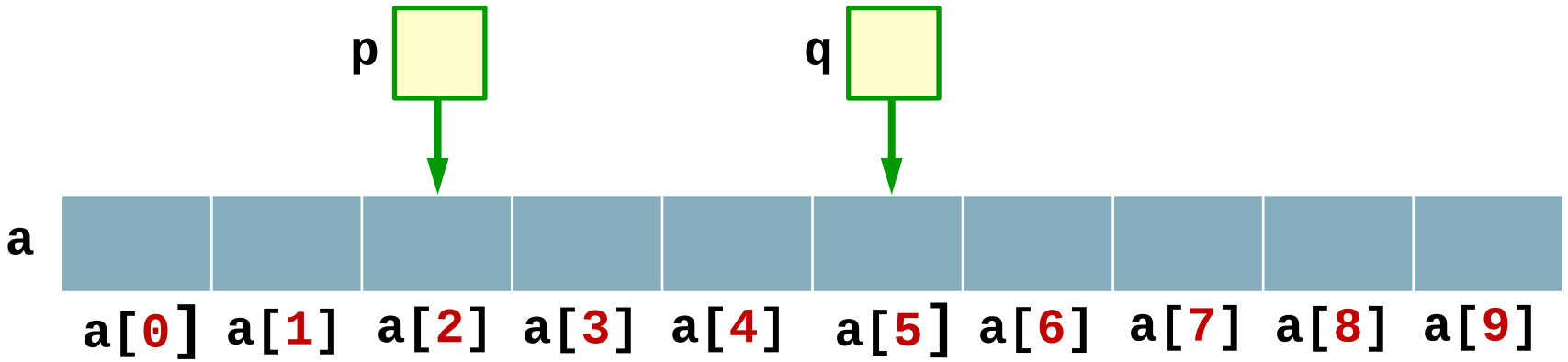
- 当指针变量指向数组元素时
  - 可以使用指针代替数组下标进行操作
- 对指针变量执行算术运算
  - 访问数组的其他元素
- 对指针可以进行哪些算术运算？
  - 加上整数，减去整数，两个指针相减



指针加上一个整数

```
int a[10];
int *p, *q;
p = &a[i];
q = p + j;
```

- 如果p指向a[i]
  - 则p+j指向a[i+j]（前提是a[i+j]必须存在）
  - p+j不是加j个字节，而是取决于p的基类型

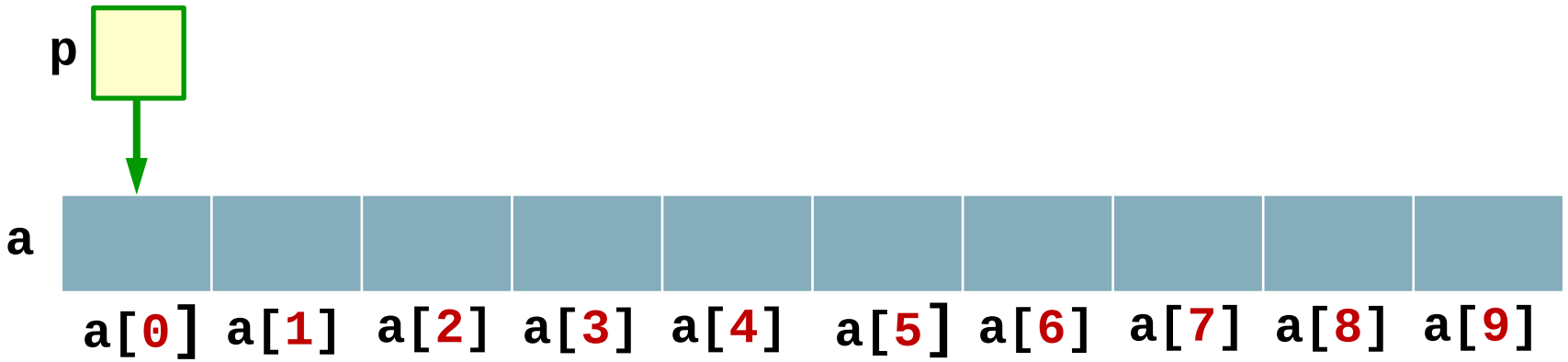


指针加上一个整数

- 指针的算术运算允许通过对指针变量**重复自增**来访问数组的元素

```
int a[10];
int *p;
```

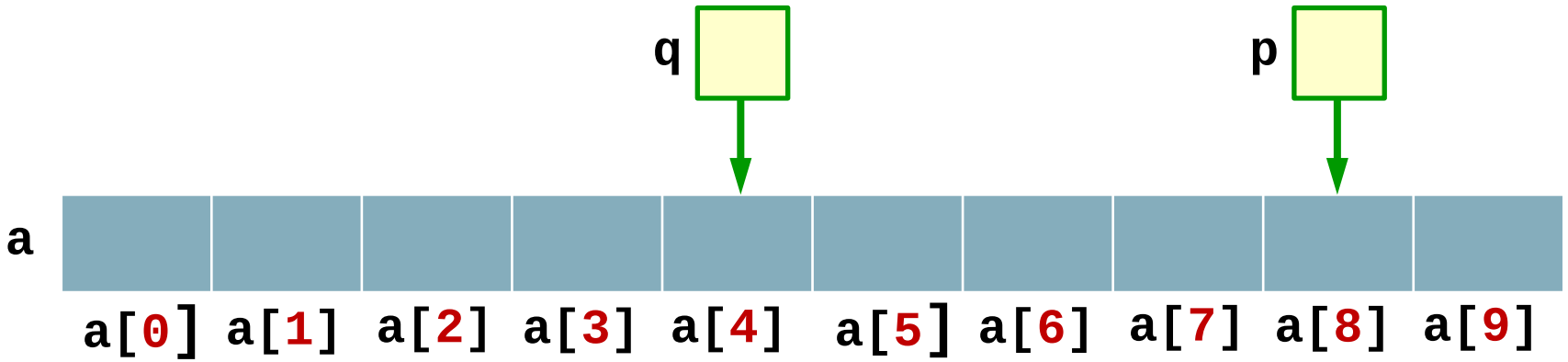
```
for (p=a; p<a+10; p++)
{
    printf("%4d", *p);
}
```



指针减去一个整数

```
int a[10];
int *p, *q;
p = &a[i];
q = p - j;
```

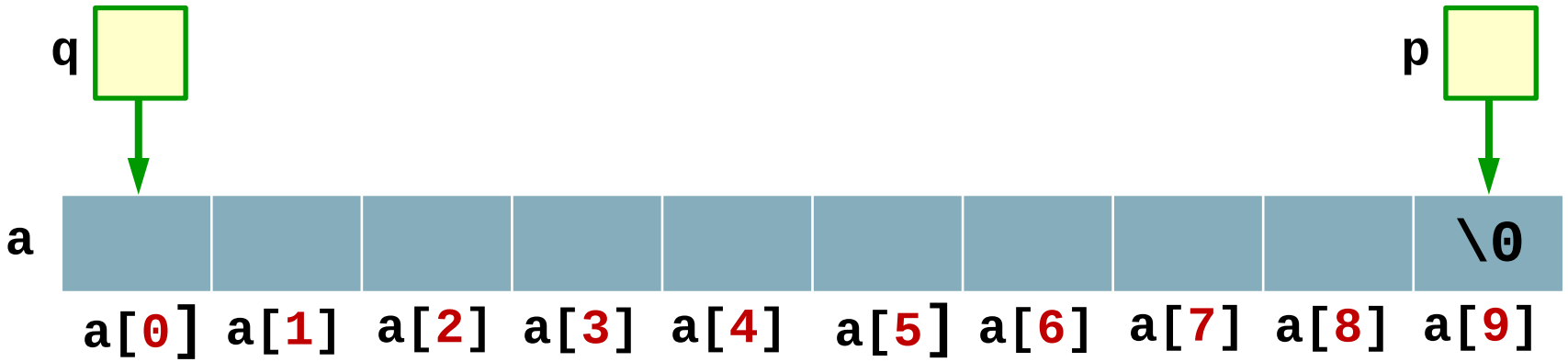
- 如果p指向a[i]
  - 那么p-j指向a[i-j]
  - 前提是a[i-j]必须存在
- 指针指向数组元素时，指针算术运算才有意义



指针相减

```
char a[10];
char *p, *q;
p = &a[i];
q = &a[j];
```

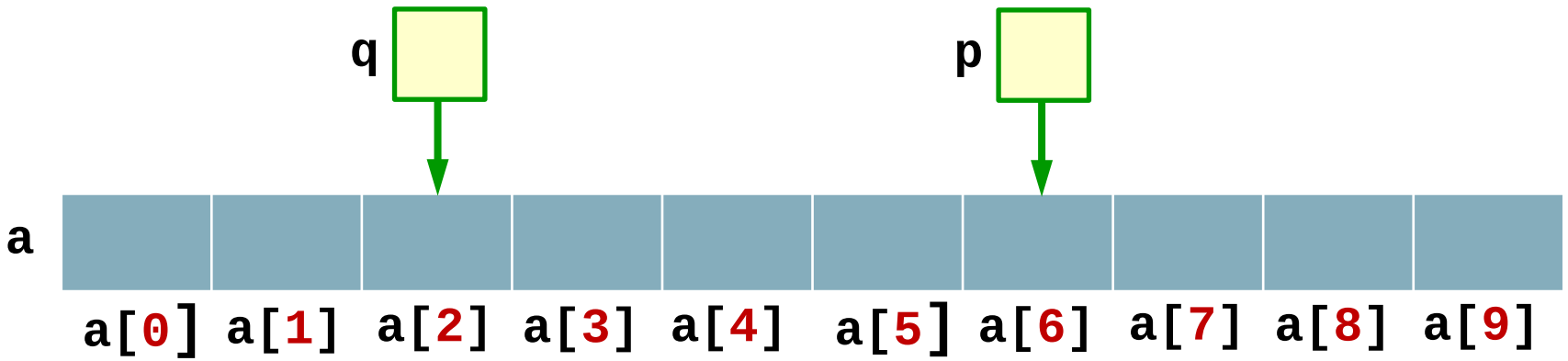
- 当两个指针相减时
  - p-q的结果为指针之间的距离i-j
  - 用来计算数组中元素的个数
- 两个指针指向同一个数组时，指针相减才有意义



指针的关系比较运算

```
char a[10];
char *p, *q;
p = &a[i];
q = &a[j];
```

- 两个指针指向同一个数组时，指针比较才有意义
  - 比较的结果依赖于数组中两个元素的相对位置
  - p >= q 为真（ q >= p 为假）



小结

- **指针变量**
  - 指针类型的变量，保存地址型数据
- **指针变量与其他类型变量的共性**
  - 在内存中占据一定大小的存储单元（通常4个字节）
  - 先定义，后使用
- **特殊性**
  - 指针变量中保存的内容只能是地址（变量或函数的地址）
  - 必须初始化后才能使用，否则指向不确定的存储单元
  - 只能指向同一基类型的变量
  - 可参与的运算：加、减整数，自增、自减、关系、赋值