为什么使用数组(Array)?

- 读入两个学生的成绩，求其平均分

```
int score1, score2;
scanf("%d", &score1);
scanf("%d", &score2);
aver = (score1 + score2) / 2;
```

- 问题：一批数据来了怎么办？

```
int score1, score2,……, score100;
scanf("%d", &score1);
scanf("%d", &score2);
……
scanf("%d", &score100);
```

```
int score, i, sum = 0;
for (i=0; i<100; i++)
{
     scanf("%d", &score);
     sum = sum + score;
}
aver = sum / 100;
```

```
int score[100], i;
for (i=0; i<100; i++)
{
    scanf("%d", &score[i]);
}
……
```

保存大量同类型的相关数据

一维数组的定义

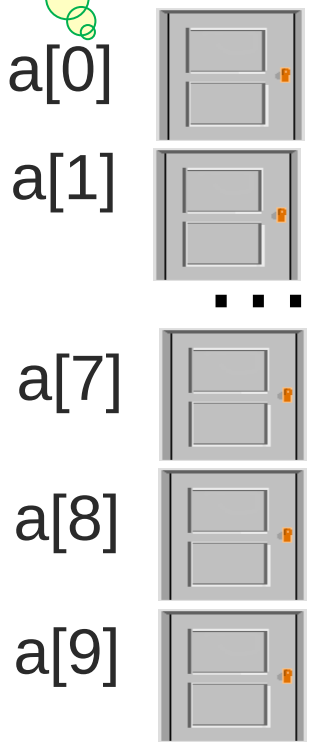
- **一维数组的定义**

```
int a[10];
```

基类型（Base Type）

数组长度

数组元素的下标从0开始
数组名a代表首地址



- **定义一个有10个int型元素的一维数组**
  - 在内存中分配连续的存储空间给此数组
- **为什么数组下标从0开始？**
  - 使编译器的实现简化一点，且下标的运算速度少量提高

一维数组的定义

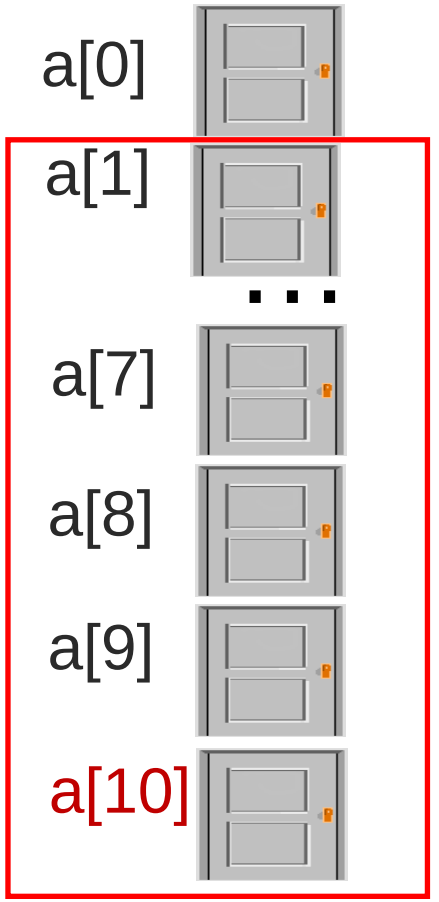
- 问题：如果希望下标从1到10而非从0到9，怎么办？

```
int a[11];
```

- `int a[n];` 
- 最好用宏定义

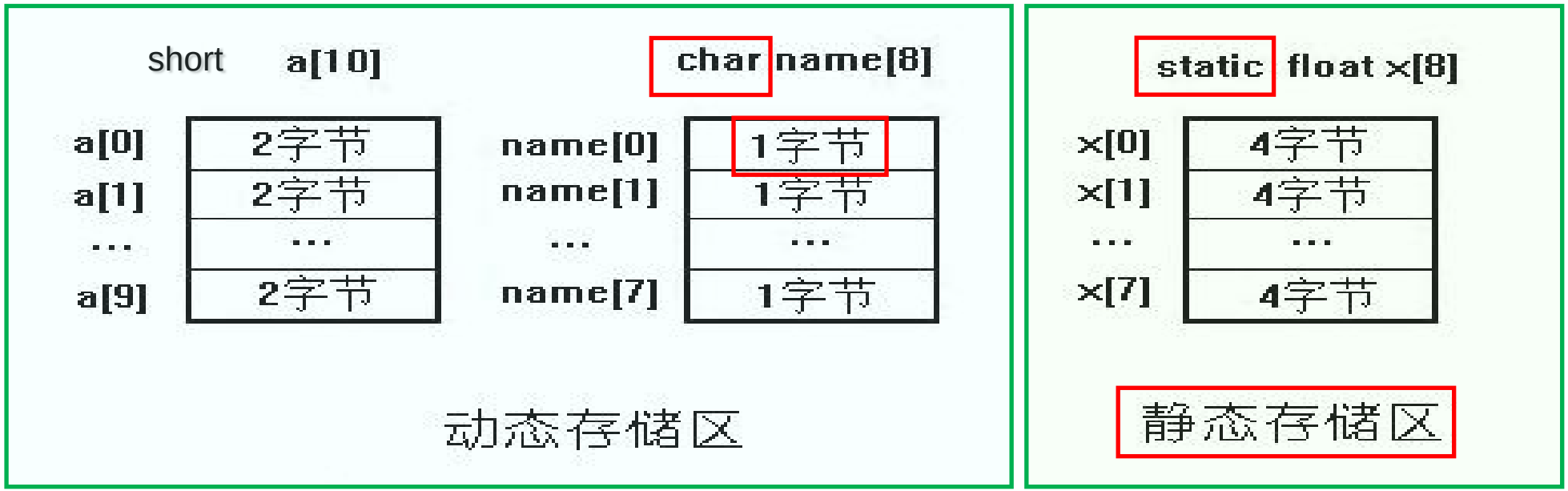
```
#define N 10
int a[N];
```

```
#define N 11
int a[N];
```



一维数组的定义

- 数组的**数据类型**——每一元素占内存空间的字节数
- 数组的**存储类型**——内存的动态、静态存储区或CPU的寄存器



一维数组在内存中占用的字节数为：数组长度× sizeof（基类型）

一维数组的初始化

- 未初始化的数组元素值是什么？
  - 静态数组和全局数组自动初始化为0值，否则，是随机数

- 一维数组的初始化

```
int a[5] = {62, 74, 56, 88, 90};
int a[5] = {62, 74};  int a[5] = {62, 74, 0, 0, 0};
int a[] = {62, 74, 56, 88, 90};
```

- 更高效的数组初始化方法

```
memset(a, 0, sizeof(a));
```

  - 用sizeof(a)来获得数组a所占的内存字节数
  - #include <string.h>

注意是string.h头文件

一维数组元素的访问

```
int a[5] = {62, 74, 56, 88, 90};
```

62	74	56	88	90
a[0]	a[1]	a[2]	a[3]	a[4]

- 一维数组的引用

  数组名[下标]

- 允许快速随机访问
  - 引用时下标允许是int型变量或表达式　　a[i]

一维数组元素的赋值

问题：如何使两个数组的值相等？

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

- 方法1:逐个元素赋值

```
b[0]=a[0];
b[1]=a[1];
b[2]=a[2];
b[3]=a[3];
```

注意又是string.h头文件

注意方向：b << a

- 方法2:通过循环语句赋值

```
int i;
for (i=0; i<4; i++)
{
    b[i] = a[i];
}
```

- 更高效的数组赋值方法

```
memcpy(b, a, sizeof(a));
```

  - 数组a复制给数组b

```
#include <string.h>
```

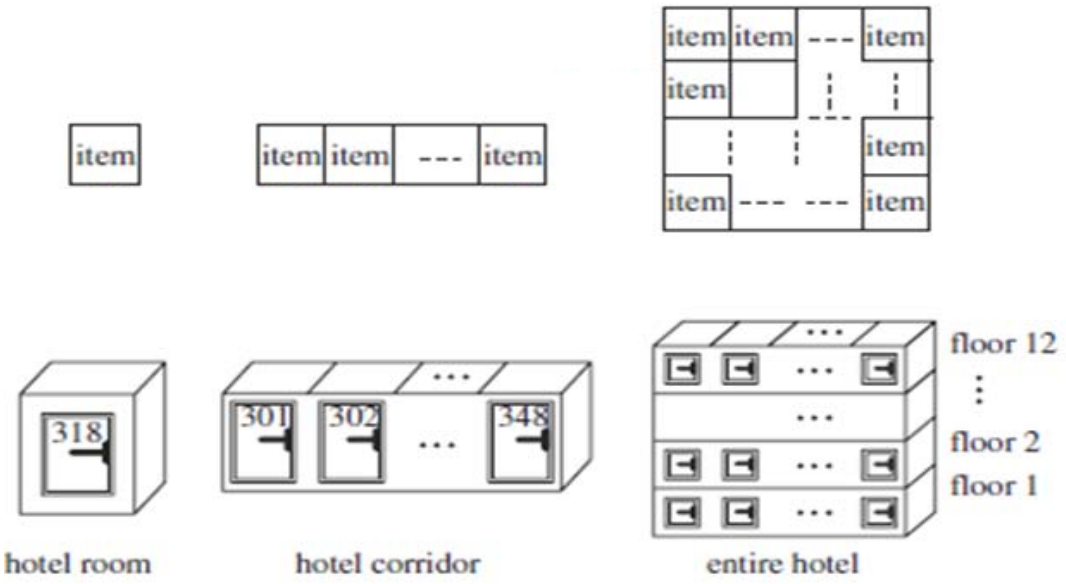
数组的逻辑存储结构

- 一维数组： int a[5];
  - 用一个下标确定元素位置

a[0]	a[1]	a[2]	a[3]	a[4]

- 二维数组: int b[2][3];
  - 用两个下标确定元素位置

b[0][0]	b[0][1]	b[0][2]
b[1][0]	b[1][1]	b[1][2]

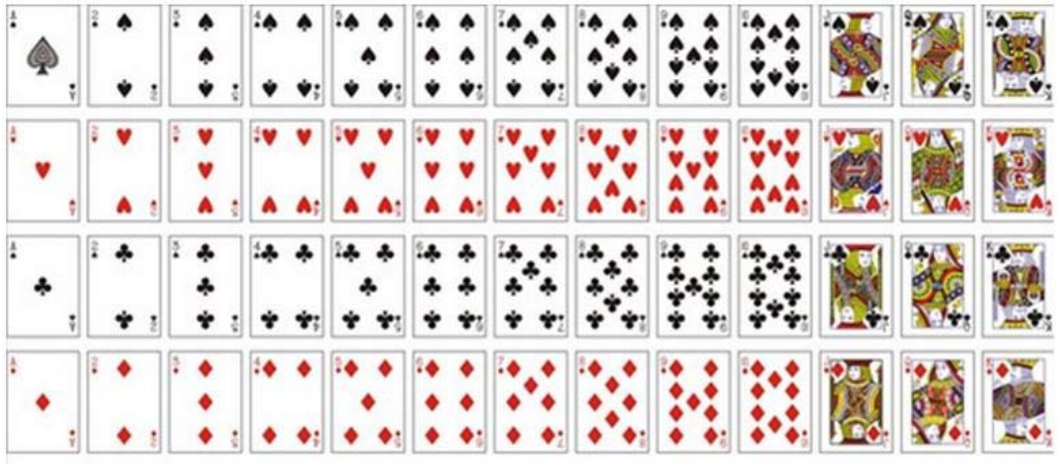


数组的逻辑存储结构

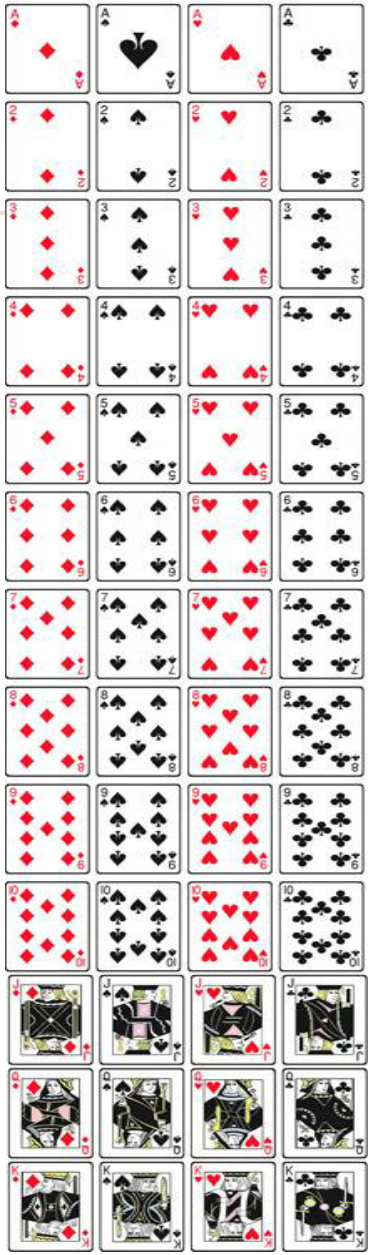
- 一维数组： int k[52];

- 三维数组：
- int k[4][4][13];
- int k[4][13][4];

- 二维数组：int k[4][13];



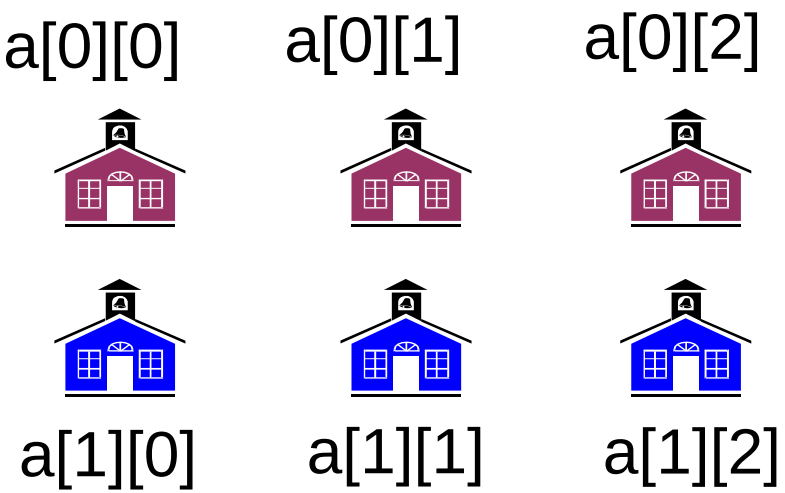
- 二维数组：int k[13][4];



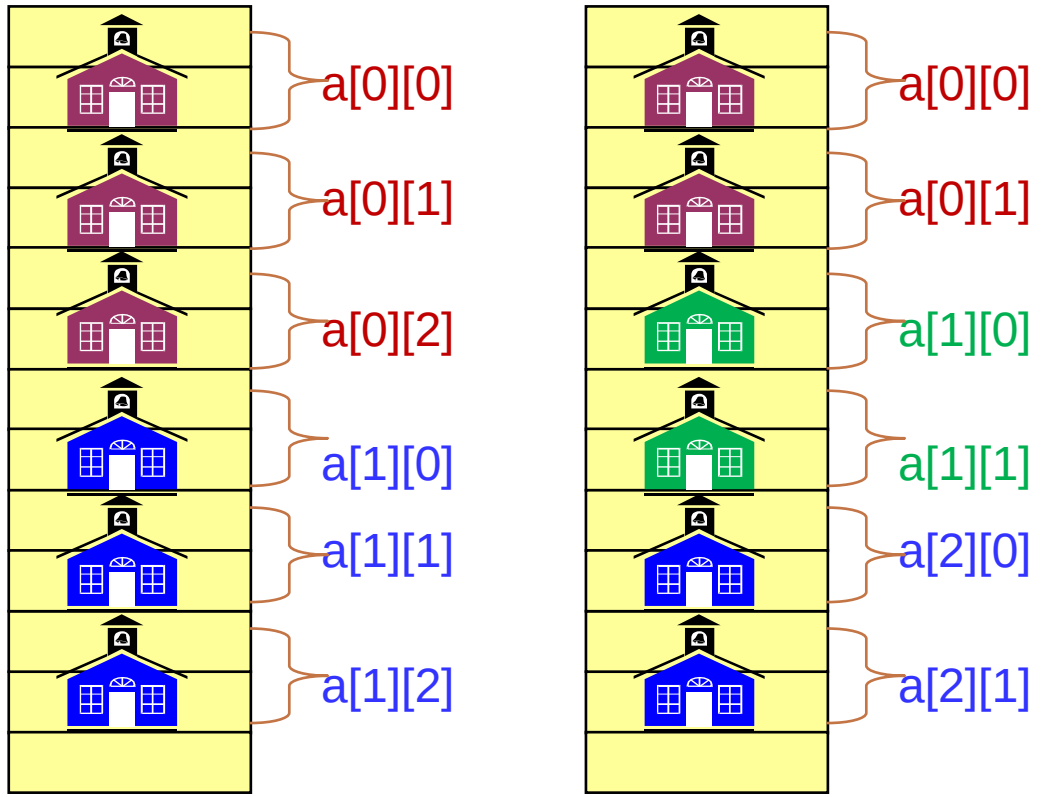
数组的物理存储结构

存放顺序：按行存放，线性存储

```
short a[2][3];
```



若short a[3][2];则…



已知每行列数才能正确读出数组元素

二维数组的定义和初始化

```
int a[ ][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};
int a[ ][4] = {{1,2,3},{4,5},{6}};
int a[3][4] = {{1,2,3,0},{4,5,0,0},{6,0,0,0}};
```

1	2	3	0
4	5	0	0
6	0	0	0

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

1	2	3	4
5	6	7	8
9	0	0	0

1	2	3
4	5	6
7	8	9

- 【例】编程实现显示用户输入的月份（不考虑闰年）拥有的天数

```
1   #include <stdio.h>
2   #define  MONTHS 12
3   int main()
4   {
5         int days[MONTHS] = {31,28,31,30,31,30,31,31,30,31,30,31};
6         int month;
7         do{
8            printf("Input a month:");
9            scanf("%d", &month);
10        }while(month < 1 || month > 12); /* 处理不合法数据的输入 */
11        printf("The number of days is %d\n", days[month-1]);
12        return 0;
13  }
```

【例】编程实现显示用户输入的月份（考虑闰年）拥有的天数

```
1   #include <stdio.h>
2   #define  MONTHS 12
3   int main()
4   {
5       int days[2][MONTHS] = {{31,28,31,30,31,30,31,31,30,31,30,31},
6                              {31,29,31,30,31,30,31,31,30,31,30,31}};
7       int year, month;
8       do{
9           printf("Input year,month:");
10          scanf("%d,%d", &year, &month);
11      } while(month < 1 || month > 12);  /* 处理不合法数据的输入 */
12      if (((year%4 == 0) && (year%100 != 0)) || (year%400 == 0))/*闰年*/
13          printf("The number of days is %d\n", days[1][month-1]);
14      else  /*非闰年*/
15          printf("The number of days is %d\n", days[0][month-1]);
16      return 0;
17  }
```

讨论

- `memcpy(b, a, sizeof(a));`
- 使用这条语句时，如果数组a和b的长度不一样，那么会导致什么结果，是否存在安全隐患？