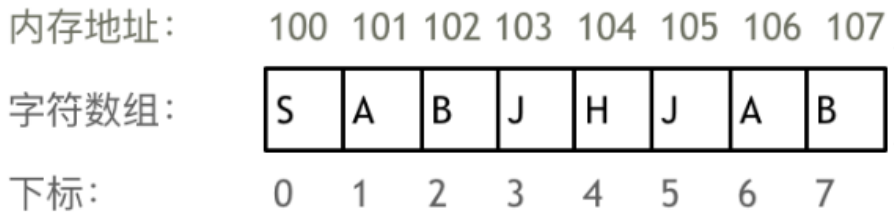
1. 数组是存放在连续内存空间上相同类型数据的集合



- 数组下标是从0开始的.
- 数组内存空间的地址是连续的;

增删数组元素要移动其它元素地址;

数组的声明:

a. 声明数组变量 arrnoob; 有10个元素的整型数组

int arrnoob[10] = {1000, ...}; or arrnoob[0]=1;
:
arrnoob[9]=10;

b. 调用元素 arrnoob[0]...

Salary = arrnoob[1];

获得数组长度:

int length = sizeof(number) / sizeof(number[0]);

总字节/元素字节;

数组名:

arrnoob代表 arrnoob数组的首地址;

如 数组名 myArray 可直接传数组

void passArray(int arr[]);

使用"&"运算符来获取数组的地址

or myArray;

int *ptr = &myArray[0]; 指针ptr被初始化为myArray的地址;

传递数组给函数:

- 指针形参: void myFunction(int *param)
- 形参已定义大小数组: void myFunction(int param[10])
- 形参未定义大小数组: void myFunction(int param[]);

函数返回数组

```c
int * getFunction(); // 只能返回静态数组名;
static int r[10];
return r;
```

2、二维数组

声明：

```c
int a[2][1] = 1; 或 int a[3][4] = {0,1,2,3,4,...,11};
int a[2][3] = 3;
```

二维数组在地址上也是连续的

输出数组：

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

静态数组与动态数组

a、静态数组 — 普通数组

声明，

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

存储在栈上；

b、动态数组 — 存储在堆上（无特定结构）

声明：

```c
int size = 5; // 先定义长度;
int *array = malloc(size * sizeof(int));
// 使用
...
// 释放内存
free(array);
```

3. 数组与指针

int runoob[] 是个数组，

其数组名为 runoob，可看作是一个指向地址 &runoob[0] 的指针

可通过 *(runoob+4) 访问数组 runoob[4]；

int *p 是一个指针变量

可通过 p = runoob；将 指针p 指向地址 &runoob[0]

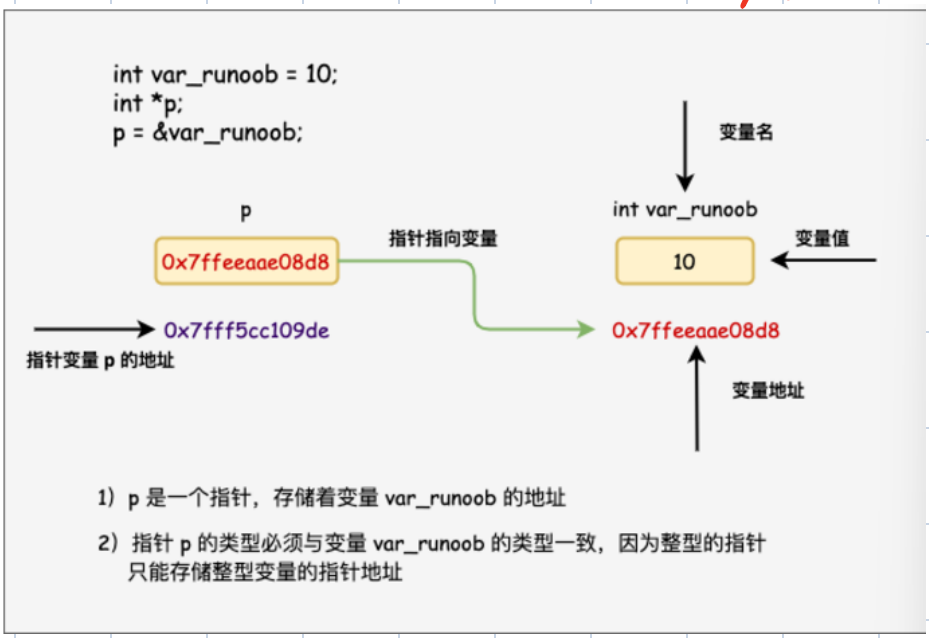
4. 指针

整型变量 int var_runoob = 10；

指针变量 int *p = NULL; // 指针只是地址；

地址操作

请用NULL来初始化指针地址



一般的指针操作： a. 定义一个指针变量；int *P;

b. 把变量地址赋给指针 p=&runoob;

or p=&runoob[2]

对于动态数组而言 int R=*p=P[2]; → c. 访问指针对应地址的值；int R=*P;

d. 打印指针所指的地址 int A=P;

e. 改变指针地址中装着的值

*ptr = A;

指针算数

```
int *ptr;

ptr++; // 指针从当前位置向后移动了4个字节;

int A = *(p+2); // 访问地址向后移2位后的值;

int A = *p+2; // 访问当前地址的值后加2;
```

指针比较：

```
ptr1 == ptr2; 是否指向同一地址;
ptr1 != ptr2; 是否指向不同地址;
ptr1 < ptr2; ptr1的地址是否在ptr2之前;
```

循环遍历：

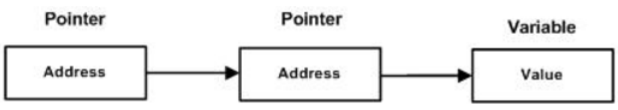
```
array[3] = {1, 2, 3};
*start = array;
*end = &array[2];
int * ptr;

for ( ptr = start; ptr<=end; ptr++)
{

};
```

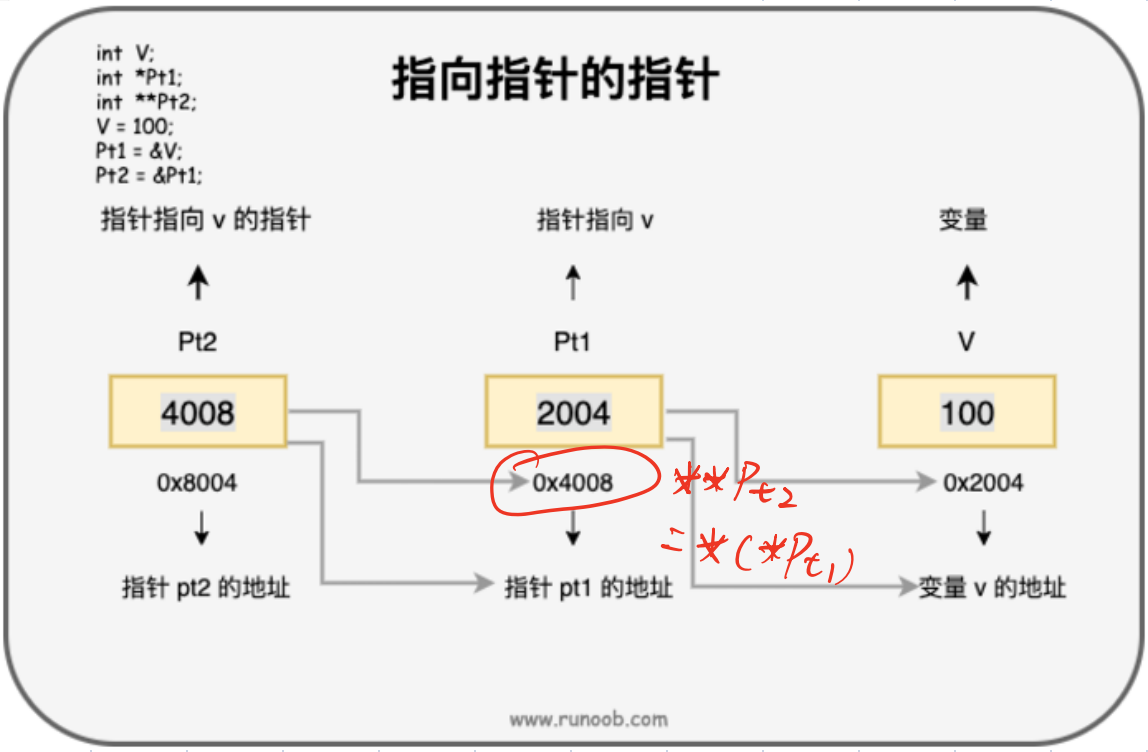
指针数组：一个由指针组成的数组；

声明 int *ptr[3];// 由3个指针组成的数组；

指向指针的指针：

声明 int ** ptr;// 用于修改数组存储地址，





5、函数指针

声明：typedef int (*fun_ptr)(int, int);

函数指针变量名　输入变量类型

回调函数：一个通过指针调用的函数

以下是来自知乎作者常溪玲的解说：

你到一个商店买东西，刚好你要的东西没有货，于是你在店员那里留下了你的电话，过了几天店里有货了，店员就打了你的电话，然后你接到电话后就到店里去取了货。在这个例子里，你的电话号码就叫回调函数，你把电话留给店员就叫登记回调函数，店里后来有货了叫做触发了回调关联的事件，店员给你打电话叫做调用回调函数，你到店里去取货叫做响应回调事件。