

首先介绍一下函数中传值与传址的概念：

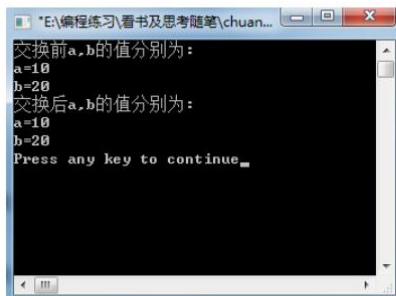
1. 传值：传值，实际是把实参的值赋值给行参，相当于copy。那么对行参的修改，不会影响实参的值。
2. 传址：实际是传值的一种特殊方式，只是他传递的是地址，不是普通的赋值，那么传地址以后，实参和行参都指向同一个对象，因此对行参的修改会影响到实参。

下来用两个例子来说明：

先看看这个代码

```
1 #include<stdio.h>
2
3 void swap(int n1,int n2)
4 {
5     int temp;
6     temp=n1;
7     n1=n2;
8     n2=temp;
9 }
10
11 int main()
12 {
13     int a=10;
14     int b=20;
15     printf("a=%d\n",a);
16     printf("b=%d\n",b);
17     swap(a,b);
18     printf("a=%d\n",a);
19     printf("b=%d\n",b);
20 }
```

以上代码实现的功能好像是交换两个数的数值对吧！运行一下看看结果：



不对啊，和我们预想的不一樣啊，可以看到a，b的值并没有被交换，怎么回事呢？

因为a和b虽然成功把值传给了n1、n2，n1、n2也完成了它们之间数值的交换，但是也仅仅是n1、n2之间交换了，和a、b没有关系。这是一次单向的传递过程，a、b能传给n1、n2，n1、n2能成功互换其数值，但n1、n2是定义在函数swap中的局部变量，当函数调用结束后，它俩就over了，被残忍抛弃了（子函数的生命期为子函数开始调用到结束调用，调用结束后就主动释放了），因此它们没有渠道把交换的值传回给a、b。所以看到的是如上图的结果。

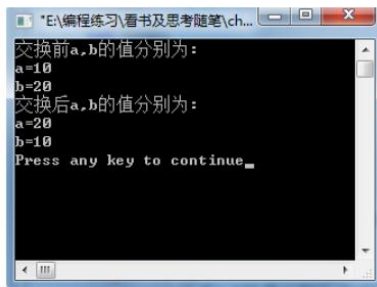
有了以上的结果，我们再来看这样一段代码：

```

1  #include<stdio.h>
2
3  void swap(int *p1,int *p2)
4  {
5      int temp;
6      temp=*p1;
7      *p1=*p2;
8      *p2=temp;
9  }
10
11 int main()
12 {
13     int a=10;
14     int b=20;
15     printf("交换前a,b的值分别为:\n");
16     printf("a=%d\n",a);
17     printf("b=%d\n",b);
18     swap(&a,&b);
19     printf("交换后a,b的值分别为:\n");
20     printf("a=%d\n",a);
21     printf("b=%d\n",b);
22 }

```

以上代码的功能同样是实现交换两个数的数值对吧！让我们再看看运行结果：



```

E:\编程练习\看书及思考随笔\ch...
交换前a,b的值分别为:
a=10
b=20
交换后a,b的值分别为:
a=20
b=10
Press any key to continue

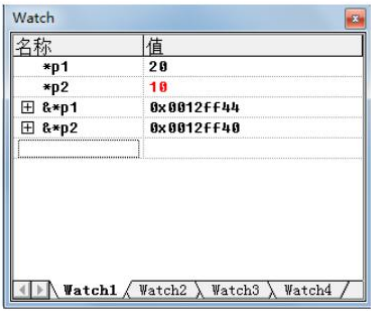
```

很奇怪，为什么这儿却能交换了？调试一下看看有什么玄机：

Watch	
名称	值
a	10
b	20
&a	0x0012FF44
&b	0x0012FF40
Watch1 Watch2 Watch3 Watch4	

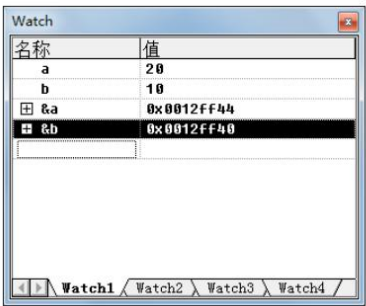
Watch	
名称	值
*p1	10
*p2	20
&*p1	0x0012FF44
&*p2	0x0012FF40
Watch1 Watch2 Watch3 Watch4	

这是调用swap函数前a、b的数值与其在内存中开辟的空间的地址以及调用函数后时*p1、*p2的数值与其地址。



The screenshot shows the 'Watch' window in a debugger. It contains a table with two columns: '名称' (Name) and '值' (Value). The table lists four items: *p1 with value 20, *p2 with value 10, &*p1 with address 0x0012FF44, and &*p2 with address 0x0012FF40. The bottom of the window has tabs labeled Watch1, Watch2, Watch3, and Watch4, with Watch1 being the active tab.

名称	值
*p1	20
*p2	10
&*p1	0x0012FF44
&*p2	0x0012FF40



The screenshot shows the 'Watch' window after the swap function call. The table now lists: a with value 20, b with value 10, &a with address 0x0012FF44, and &b with address 0x0012FF40. The &b row is highlighted. The tabs at the bottom remain the same.

名称	值
a	20
b	10
&a	0x0012FF44
&b	0x0012FF40

这是调用swap函数后a、b的数值与其在内存中开辟的空间的地址以及开始调用函数时*p1、*p2的数值与其地址。

可以看到此时a、b与*p1、*p2的地址空间是一样的，那么当*p1、*p2被修改时，a、b也会跟着发生变化，因为此时二者占用了同一块空间，当任意一者使空间里的内容发生变化时，二者都会做相同变化。

举个不太恰当的例子，把夫妻二人各看做一个变量，把它们的共用的银行卡看做它们占用的同一块空间，此时，他俩拥有的财产是一样的，都是银行卡里的钱，那么，不管谁花了或存了钱，他两的财产都会发生同等改变，任意一人对财产的修改会影响到另一个人的财产。（当然，私房钱不算）。仔细想想，其实大概就是这么个道理。

函数调用的传值与传址大概就是这么个过程，这块内容其实不难理解，只要知道其概念，通过实验验证，很容易掌握。