函数的传值调用

阅读下面的程序，观察输出结果。

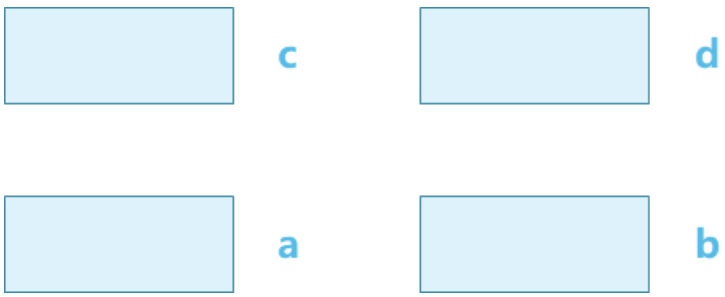
```
#include<iostream>
using namespace std;
void swap(int a, int b)
{
      int tmp = a;
      a = b;
      b = tmp;
}
int main()
{
    int c = 1;
    int d = 2;
    swap(c, d);
    cout << c << ' ' << d << endl;
    return 0;
}
```

【输出结果】

```
1 2
```

【分析】

这种调用方式是将实参的数据值传递给形参，即将实参值拷贝一个副本传递给被调用函数。在被调用函数中，形参值可以改变，但不影响主调函数的实参值。参数传递方向只是从实参到形参，简称单向值传递。



所以，在此例中，虽然在 swap 函数中交换了 a、b 两数的值，但是在 main 中却没有交换。因为 swap 函数只是交换 c、d 两变量副本的值，实参值没有改变，并没有达到交换的目的。

函数的引用调用

阅读下面的程序，观察输出结果。

```
#include<iostream>
using namespace std;
void swap(int &a, int &b)
{
      int tmp = a;
      a = b;
      b = tmp;
}
int main()
{
    int c = 1;
    int d = 2;
    swap(c, d);
    cout << c << ' ' << d << endl;
    return 0;
}
```

【输出结果】

```
2 1
```

【分析】

这种使用引用传递函数的参数的调用方法，在内存中并没有产生实参的副本，它是直接对实参操作；这就提供了一种可以改变实参变量的值的方法。

在此例中，因为 swap 函数的参数为引用调用，所以，在函数 swap 中修改 a、b 的值相当于在主函数 main 中修改 c、d 的值。