指標

5/3, hydai

指標？

- 在 File IO 出現過 **FILE *file;**
- 還記得這個 * 符號嗎？
- 如果我們能從 file 讀取資料，那麼 file 本身是？

```c
FILE *file = fopen("file.txt", "r");
char str[1000];
printf("%p\n", file);
fgets(str, 100, file);
printf("%s", str);
```

第一個 printf 會印出 file 本身的值
第二個 printf 會印出從 file 讀出來的字串

```
➜  Pointer git:(master) ✗ ./a.out
2062365344
Hello, world
```

指標變數 Pointer

- 專門用來儲存位址
- 宣告的方式： type * name;
- type 會根據你要存的形態有關
  - 要指向 int 的指標用 int * name;
  - 要指向 char 的指標用 char * name;

位址？？

- 任何變數到底存在哪裡？
- 應該有個位址讓人尋找？
- 那個位址怎麼找呢？

```c
int i = 100;
printf("i = %d\n", i);
printf("address of i = %p\n", &i);
```

第一個 printf 印出 i 裡面的值
第二個 printf 印出 i 這個變數實際的位址

```
➜  Pointer git:(master) ✗ ./a.out
i = 100
address of i = 1467438508
```

指標……指向誰？

```
int i = 100;
int *ptr = &i;
printf("i = %d\n", i);
printf("address of i =%p\n", &i);
printf("ptr =%p\n", ptr);
printf("value which ptr point to = %d\n", *ptr);
```

```
➜  Pointer git:(master) ✗ ./a.out
i = 100
address of i = 1585640876
ptr = 1585640876
value which ptr point to = 100
```

int i = 100;

Address 1585640876

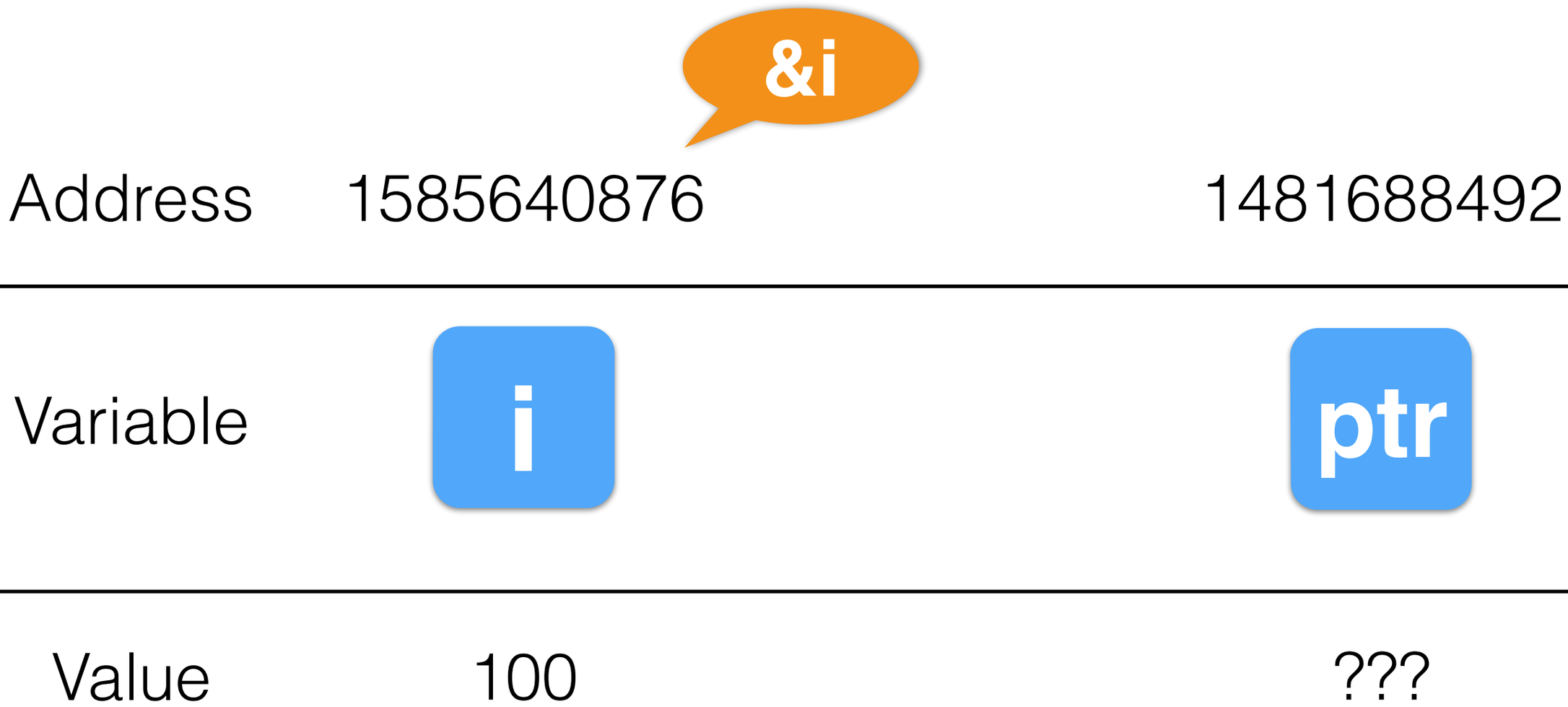
Variable

Value 100

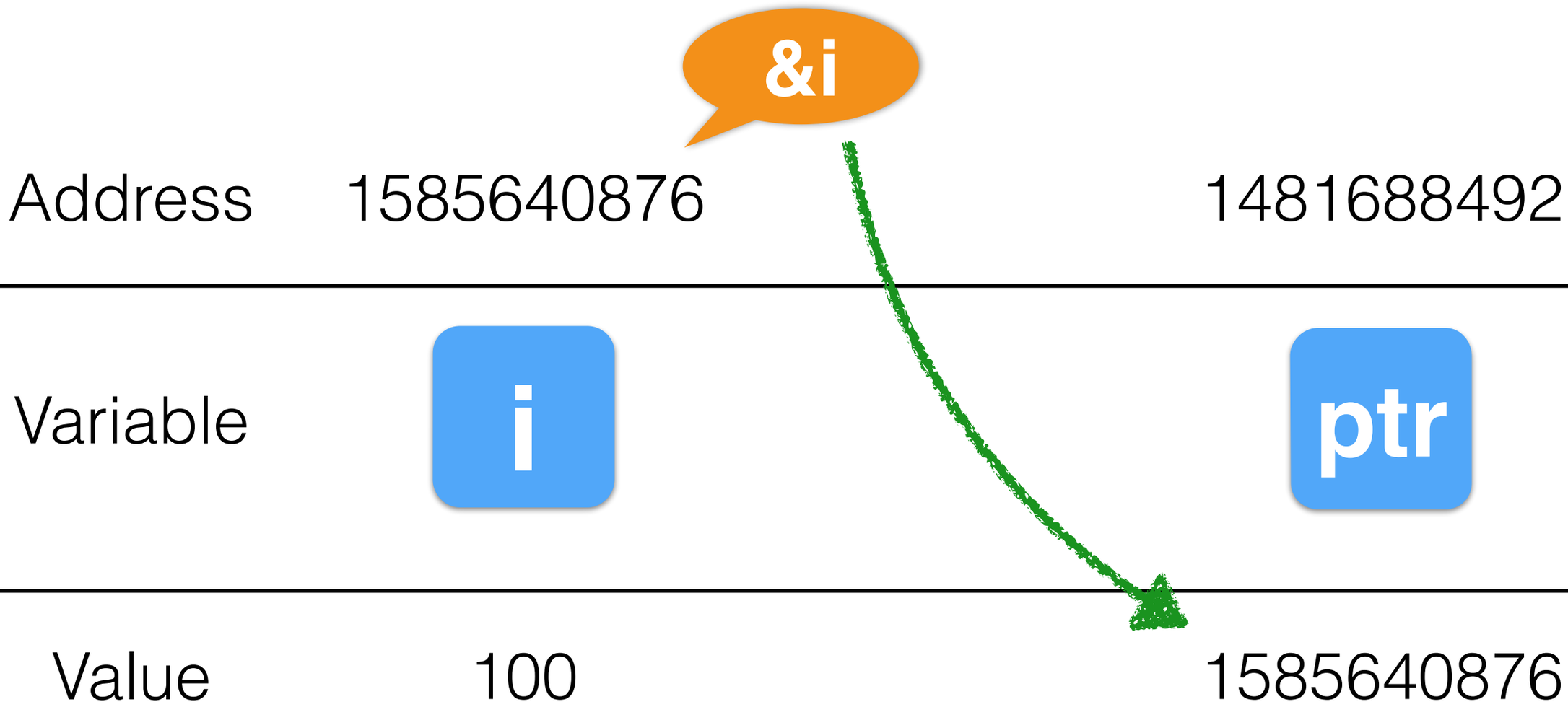
int *ptr;

Address	1585640876	1481688492
Variable	i	ptr
Value	100	???

&i ——> 取得變數 i 的位址



ptr = &i;


&i
Address
1585640876
1481688492
Variable
i
ptr
Value
100
1585640876

Address	1585640876	1481688492
Variable	i	ptr
Value	100	1585640876

一些操作符號

- var 是一個變數, ptr 是一個指標變數
- var => 變數的值
- &var => 取得變數的位址
- ptr => 指標變數的值
- *ptr => 取得指標變數指到的值

範例：數字交換

- 利用函式把整數 a, b 交換
- Input:
  - 123 100
- Output:
  - 100 123

猜猜看結果？

```
1 #include <cstdio>
2 void swap(int x, int y) {
3     int tmp = x;
4     x = y;
5     y = tmp;
6 }
7 int main(int argc, char *argv[])
8 {
9     int a = 221, b = 124;
10    printf("Before swap, a = %d, b = %d\n", a, b);
11    swap(a, b);
12    printf("After swap, a = %d, b = %d\n", a, b);
13    return 0;
14 }
```

完全沒有換！

```
➜  Pointer git:(master) ✗ ./a.out
Before swap, a = 221, b = 124
After swap, a = 221, b = 124
```

換個寫法

```
1 #include <cstdio>
2 void swap(int *x, int *y) {
3     int tmp = *x;
4     *x = *y;
5     *y = tmp;
6 }
7 int main(int argc, char *argv[])
8 {
9     int a = 221, b = 124;
10    printf("Before swap, a = %d, b = %d\n", a, b);
11    swap(&a, &b);
12    printf("After swap, a = %d, b = %d\n", a, b);
```

成功了！

```
➜  Pointer git:(master) ✗ ./a.out
Before swap, a = 221, b = 124
After swap, a = 124, b = 221
```

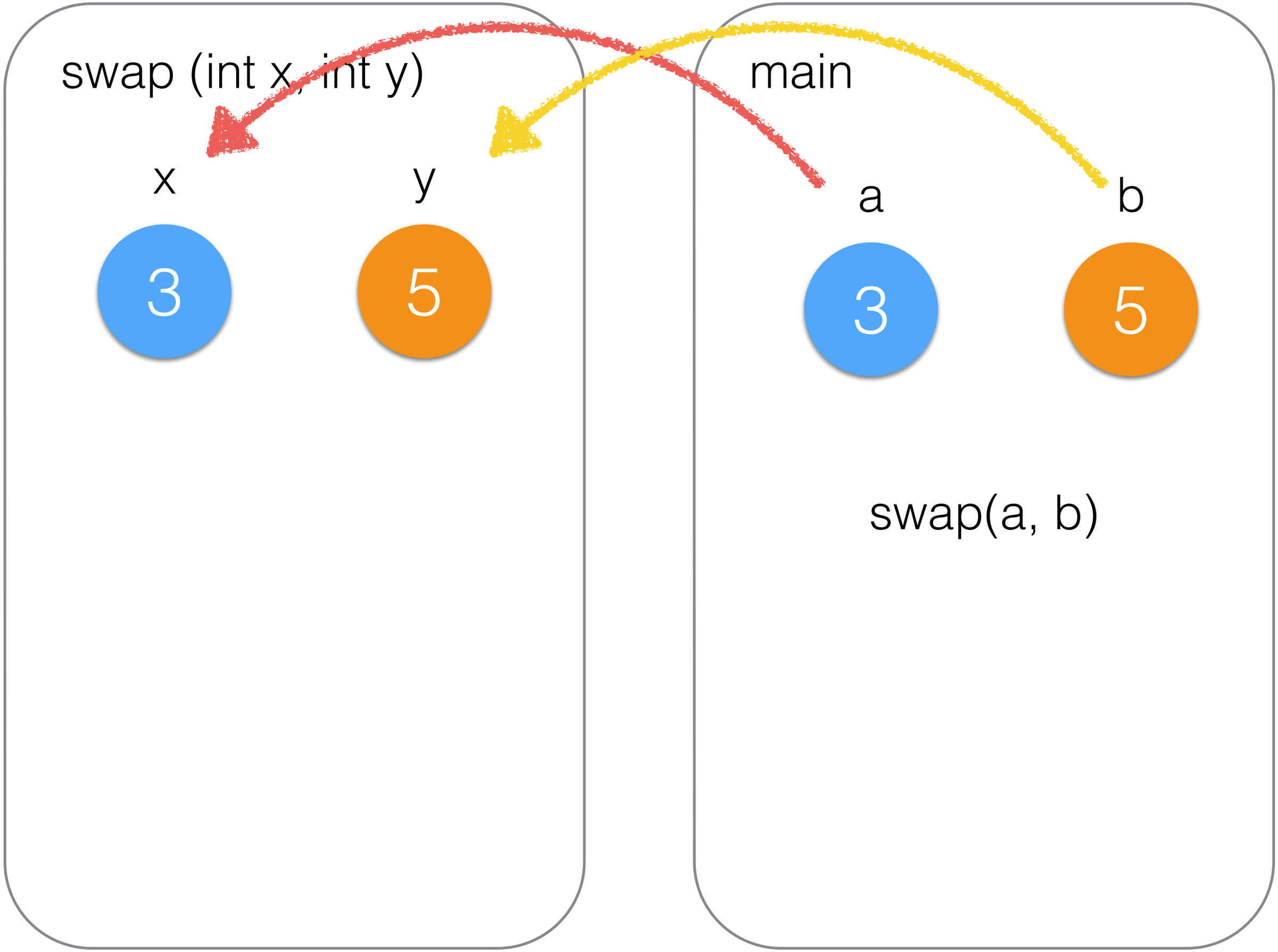
為什麼？

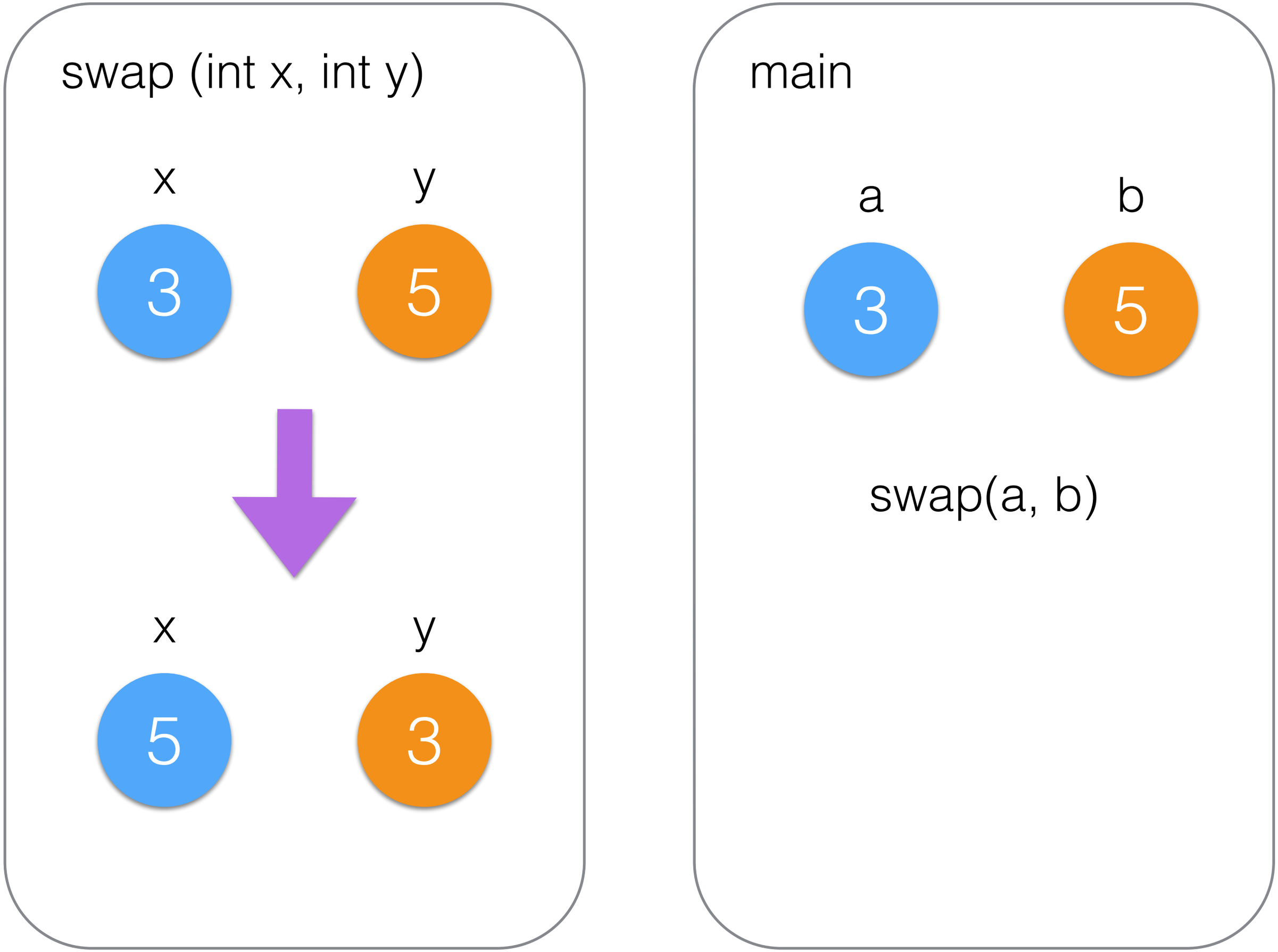
- 函式只是將傳進來的值複製一份
  - 因此交換的時候，是把複製的那一份換掉
- 如果用指標，是把指到的東西交換
  - 如此就能成功換了！

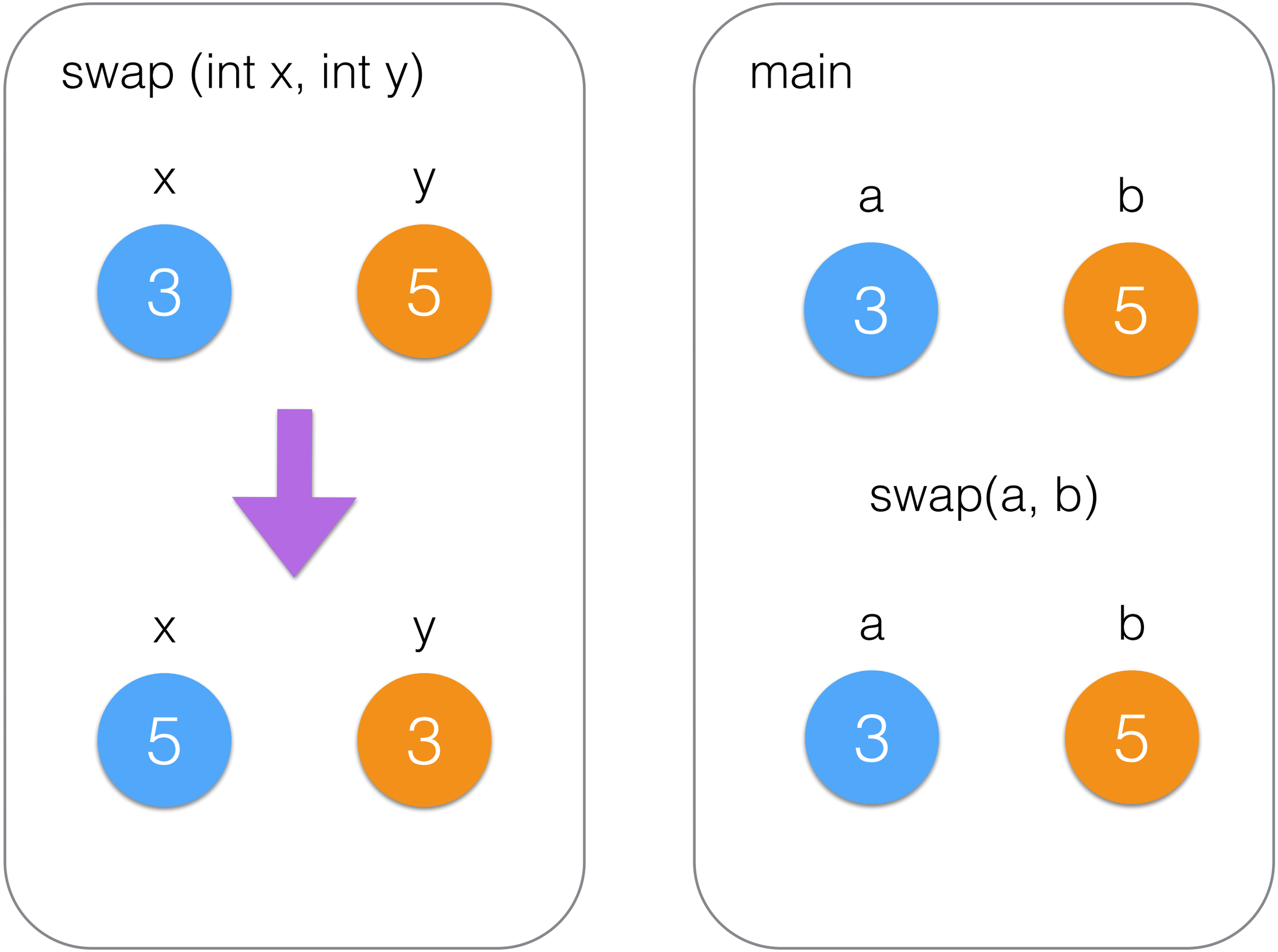
swap (int x, int y)

main

swap(a, b)


swap (int x, int y)
x
y
3
5
main
a
b
3
5
swap(a, b)


swap (int x, int y)
x
y
3
5
x
y
5
3
main
a
b
3
5
swap(a, b)


swap (int x, int y)
x
y
3
5
x
y
5
3
main
a
b
3
5
swap(a, b)
a
b
3
5

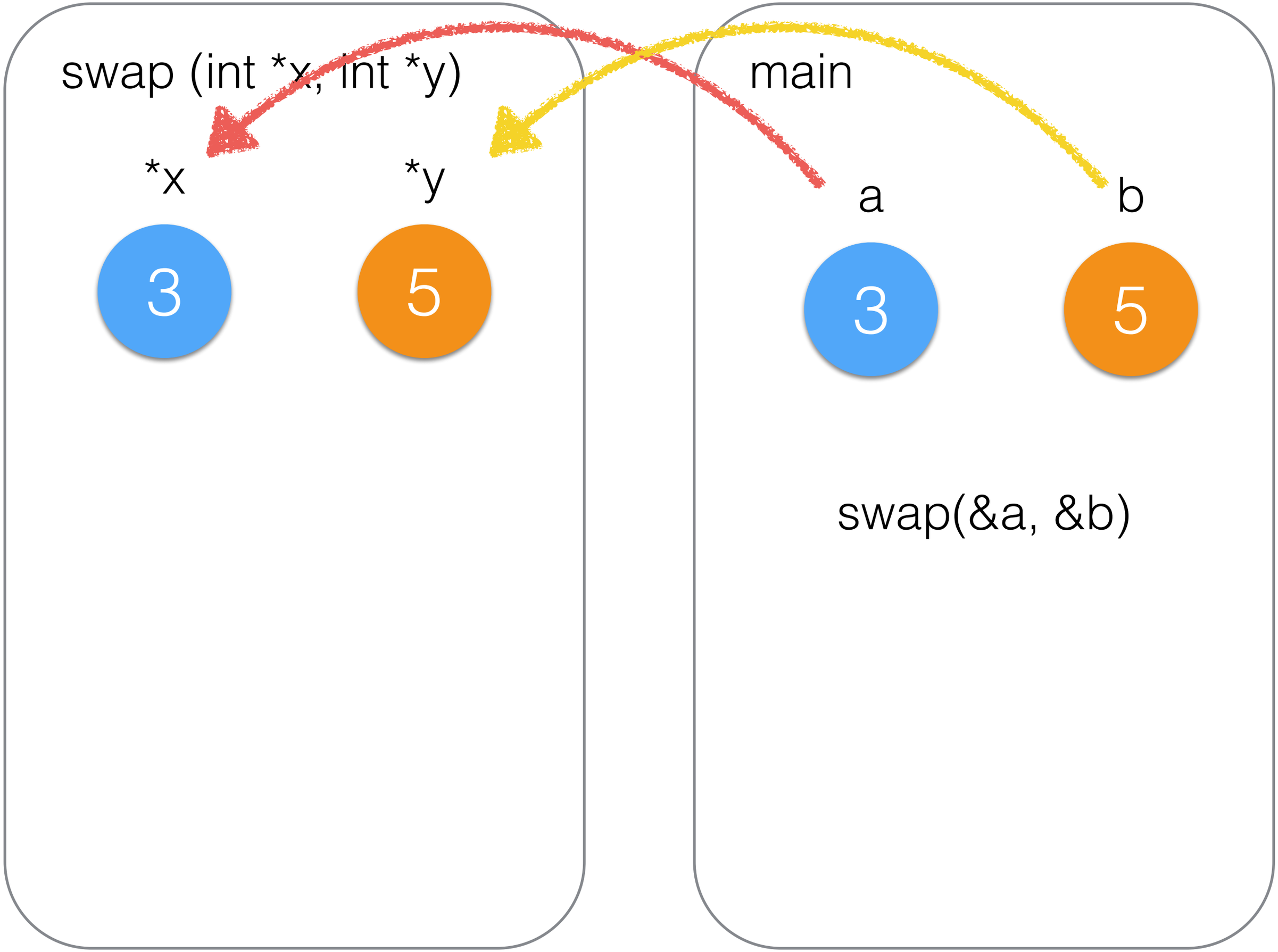
swap (int *x, int *y)

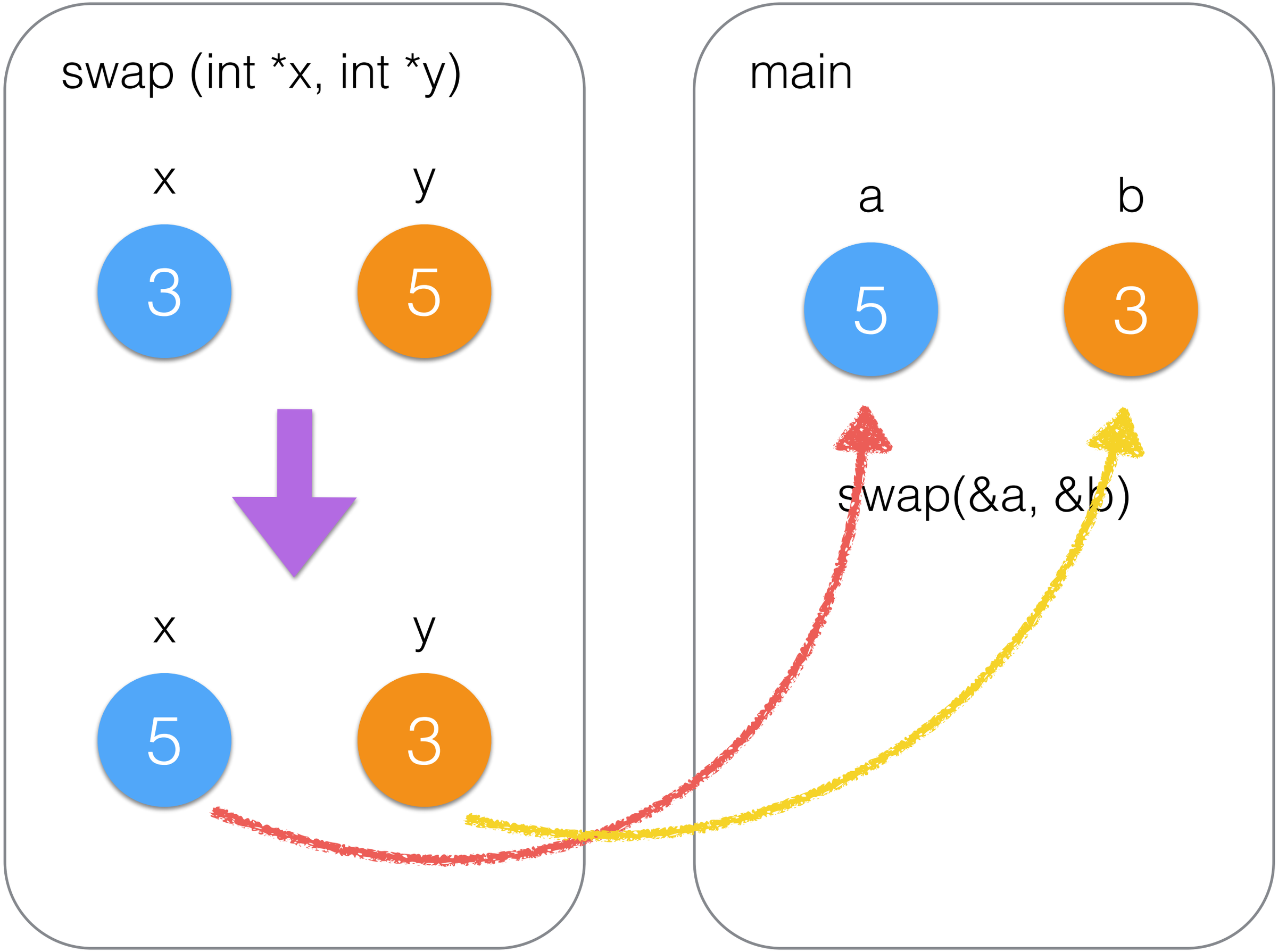
main

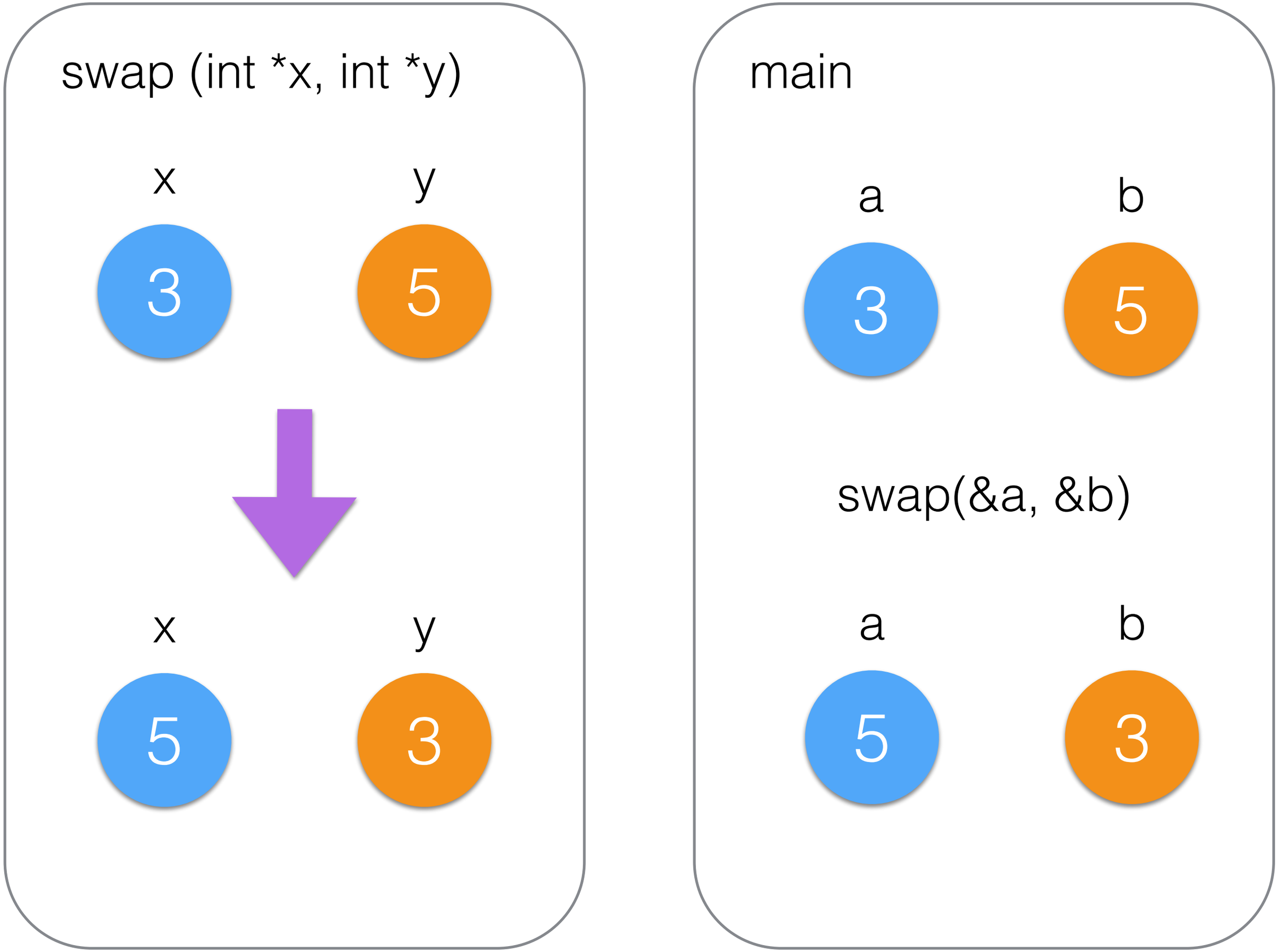
a

b

swap(&a, &b)


swap (int *x, int *y)
*x
*y
3
5
main
a
b
3
5
swap(&a, &b)


swap (int *x, int *y)
x
y
3
5
x
y
5
3
main
a
b
5
3
swap(&a, &b)


swap (int *x, int *y)
x
y
3
5
x
y
5
3
main
a
b
3
5
swap(&a, &b)
a
b
5
3

Bubble Sort

- 將 Bubble Sort 抽成一個 function
- 只要送 array 和 array 的 size 就可以排序
- void bubble_sort(int array[], int size);

原本的 bubble sort

```
int arr[10] = {3, 1, 2, 4, 5, 6, 8, 6, 9, 1};
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10 -1 - i; j++) {
        if (arr[j] > arr[j+1]) {
            int tmp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = tmp;
        }
    }
}
```

抽離後 bubble sort

```
void bubble_sort(int array[], int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size - 1 - i; j++) {
            if (array[j] > array[j+1]) {
                int tmp = array[j];
                array[j] = array[j+1];
                array[j+1] = tmp;
            }
        }
    }
}
```

```
bubble_sort(arr, 10);
```

為什麼交換了？

- 如果傳 array 進去也是拷貝一份的話.....
- 那為什麼原本的 array 也會變？
- 這肯定有什麼誤會？！！！

陣列也是一種指標

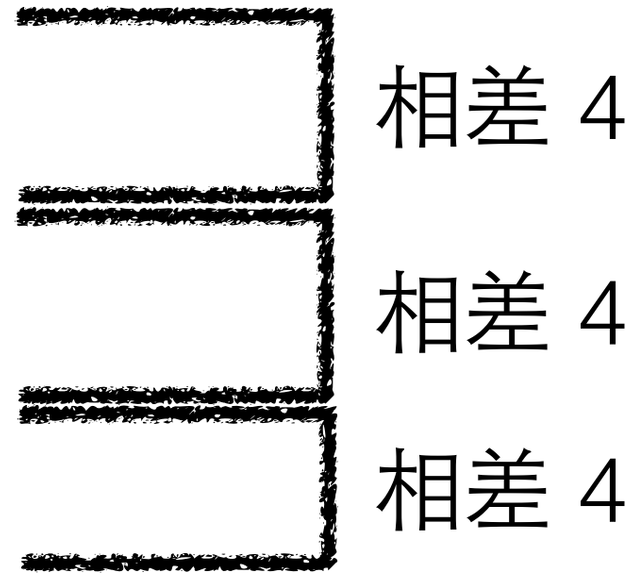
- 於是，能傳東西進去，改了還會變
- 我們可以說陣列的頭也算是一種指標

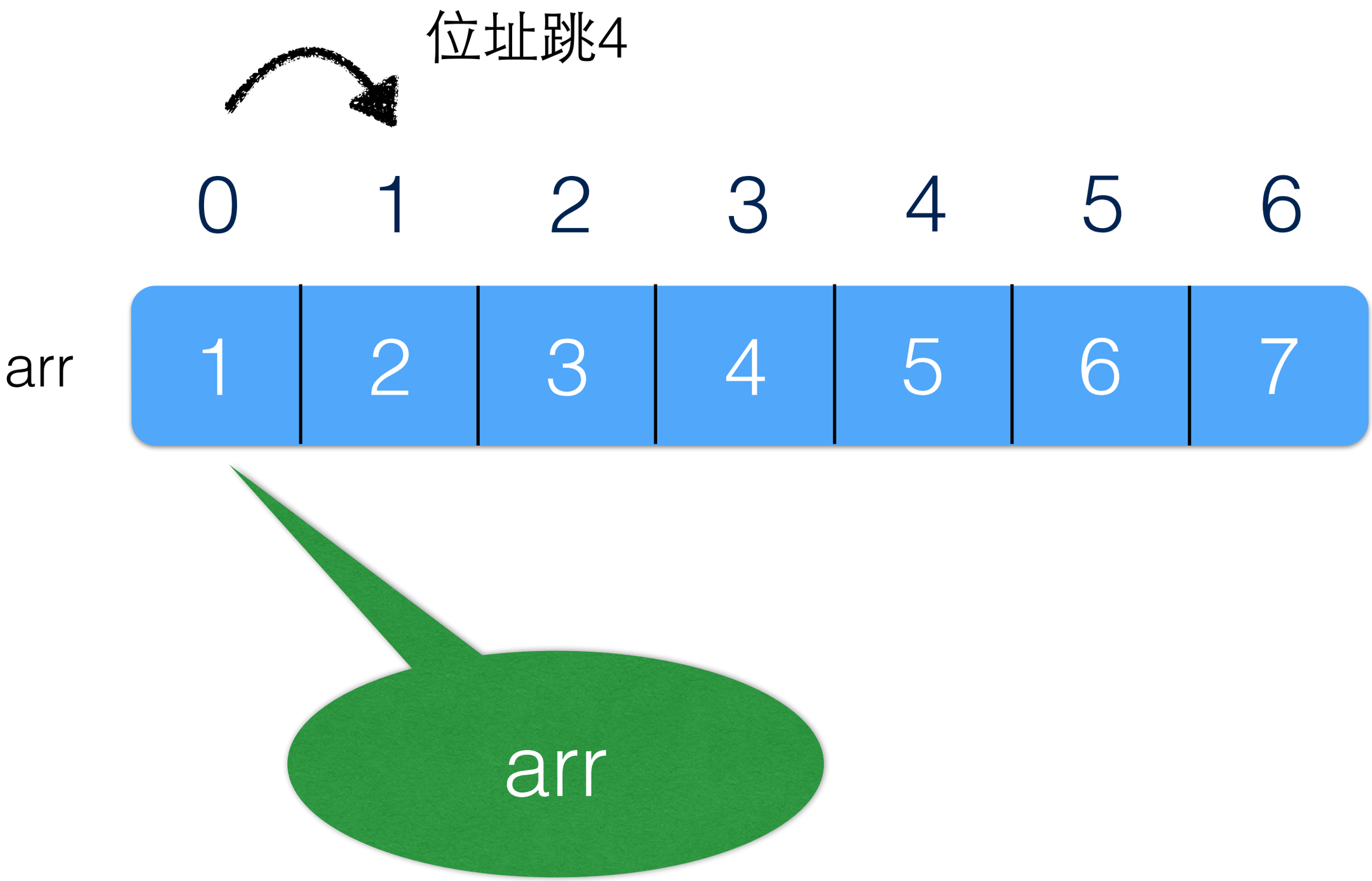
印印看位址

```
int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
printf("arr = %p\n", arr);
printf("arr[0] = %d\n", arr[0]);
printf("address of arr[0] = %p\n", &arr[0]);
```

```
➜  Pointer git:(master) ✗ ./a.out
arr = 1537824224
arr[0] = 1
address of arr[0] = 1537824224
```

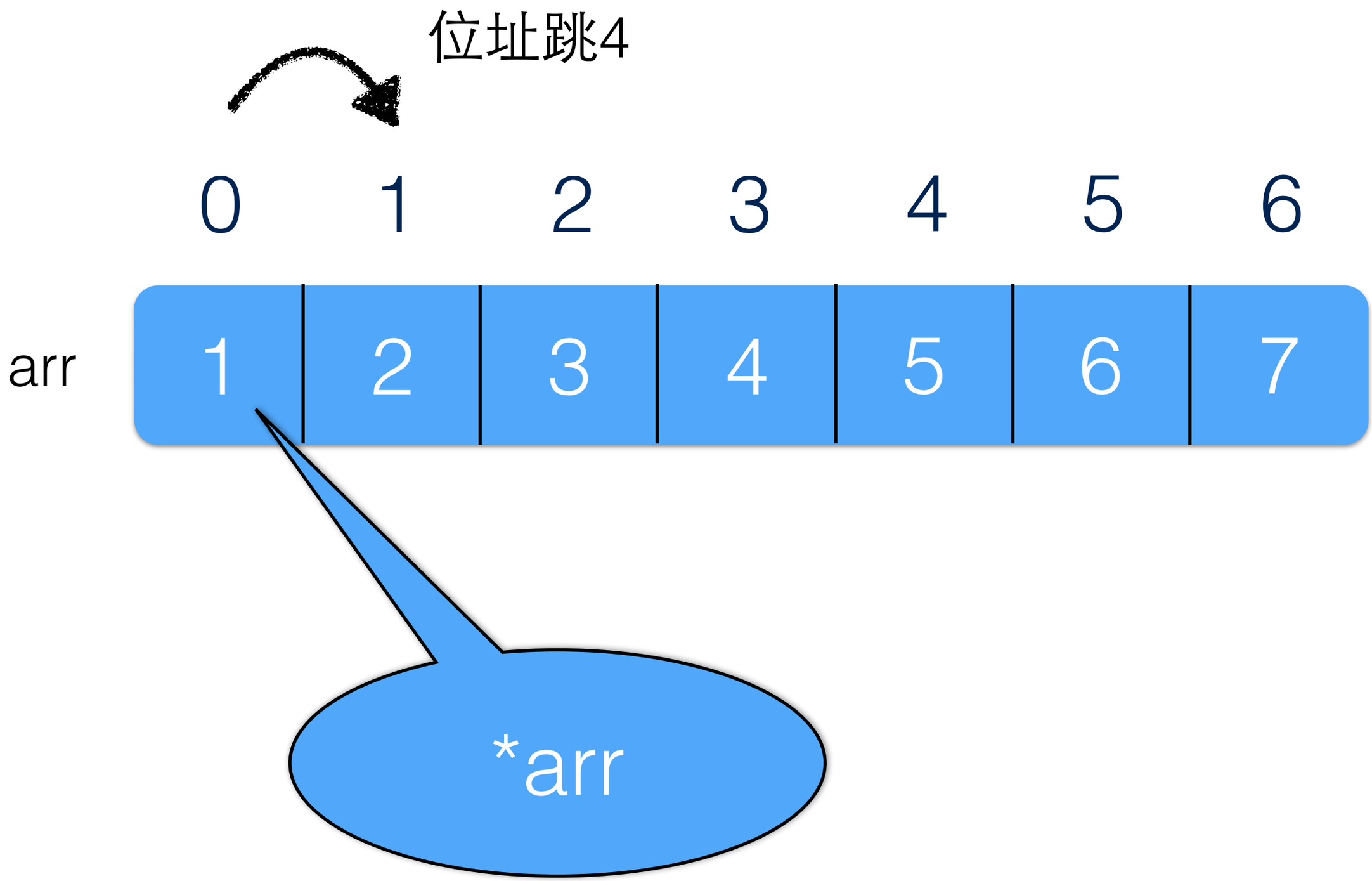
```
➜  Pointer git:(master) ✗ ./a.out
arr = 1510606304
arr[0] = 1
address of arr[0] = 1510606304
arr[1] = 2
address of arr[1] = 1510606308
arr[2] = 3
address of arr[2] = 1510606312
arr[3] = 4
address of arr[3] = 1510606316
arr[4] = 5
address of arr[4] = 1510606320
arr[5] = 6
address of arr[5] = 1510606324
arr[6] = 7
address of arr[6] = 1510606328
arr[7] = 8
address of arr[7] = 1510606332
arr[8] = 9
address of arr[8] = 1510606336
arr[9] = 10
address of arr[9] = 1510606340
```




位址跳4
0
1
2
3
4
5
6
arr
1
2
3
4
5
6
7
arr

既然位址相同……

- 還記得剛才提到的對指標取值嗎？
- 我們來試試看對 arr 進行取值
- *arr = ?


位址跳4
0
1
2
3
4
5
6
arr
1
2
3
4
5
6
7
*arr

對應關係

- arr <=> &arr[0]
- *arr <=> arr[0]
- 那.....arr[1], arr[2], ……, arr[n] 呢？

再印印看

```
int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
printf("arr = %d\n", arr);
for (int i = 0; i < 10; i++) {
    printf("arr[%d] = %d\n", i, arr[i]);
    printf("*(arr+%d) = %d\n", i, *(arr+i));
    printf("&arr[%d] =%p\n", i, &arr[i]);
    printf("arr+%d= %p\n\n", i, arr+i);
}
```

```
➜  Pointer git:(master) ✗ ./a.out
arr = 1402246624
arr[0] = 1
*(arr+0) = 1
&arr[0] = 1402246624
arr+0 = 1402246624

arr[1] = 2
*(arr+1) = 2
&arr[1] = 1402246628
arr+1 = 1402246628

arr[2] = 3
*(arr+2) = 3
&arr[2] = 1402246632
arr+2 = 1402246632

arr[3] = 4
*(arr+3) = 4
&arr[3] = 1402246636
arr+3 = 1402246636

arr[4] = 5
*(arr+4) = 5
&arr[4] = 1402246640
arr+4 = 1402246640
```

arr[i] <=> *(arr+i)

&arr[i] <=> (arr+i)

為什麼每次都加四？

- 有發現到嗎？
- 我寫 arr+1 的時候他卻是跳四！
- 對 &arr[1], &arr[0] 之間也是跳四！

談談 int

- int => 32-bit signed number
- 32 bits => 4 bytes (1 bytes = 8bits)
- memory unit is 1 byte
- 1 int => 4*1 byte => 4 units

可是他怎麼知道跳四？

- 編譯器會去看你指標的形態
- 如果是 char * 就只會跳一
- 如果是 int * 就會跳四

練習：strlen

- 給一個 string ，請輸出他的長度
- 寫出 strlen 的函式
- 結尾用 ‘\0’ 來檢查