

CS 161, Lecture 14: Different Ways to Pass Parameters

How We Have Been Passing -> By Value

- Also referred to as Call by Value
- Copies the value into the formal parameter

```
void swap (int a, int b) {
```

```
    int temp = a;
```

```
    a = b;
```

```
    b = temp;
```

```
}
```

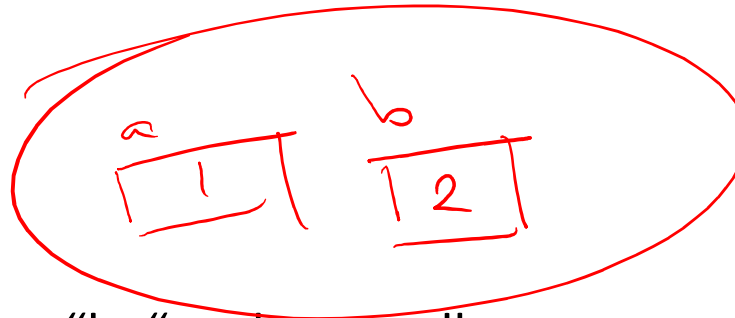
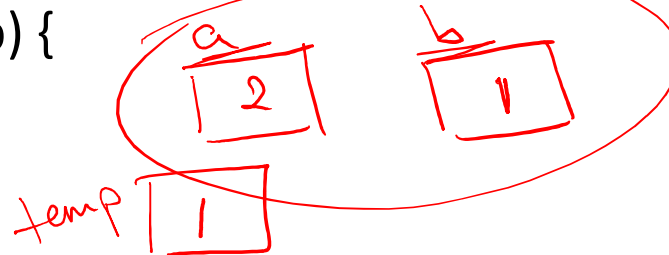
```
int main () {
```

```
    int a = 1, b = 2;
```

```
    swap(a, b);
```

```
    cout << "a: " << a << "b: " << b << endl;
```

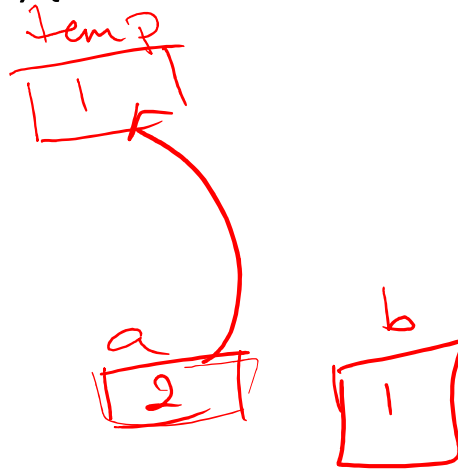
```
}
```



Alternate: Pass By Reference

- Takes both the value and the address of the passed in variable
- Does not exist in C
- References can't be null

```
void swap (int &a, int &b) {  
    int temp = a;  
    a = b;  
    b = temp;  
}  
int main () {  
    int a = 1, b = 2;  
    swap(a, b);  
    cout << "a: " << a << "b: " << b << endl;  
}
```



The diagram illustrates the state of memory during the execution of the swap function. It shows three memory locations: 'temp' (containing 1), 'a' (containing 2), and 'b' (containing 1). A red arrow points from the 'temp' box to the 'a' box, indicating that the value of 'temp' is being assigned to 'a'.

Alternate: Pass By Pointer

- Pointer is a memory address
- Can be changed to hold different memory addresses
- Pointers need to be dereferenced to get to the value stored at that address

```
void swap (int* a, int* b) {  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

→ pointer

→ dereference operator

```
int main () {  
    int a = 1, b = 2;  
    swap(&a, &b);  
    cout << "a: " << a << "b: " << b << endl;  
}
```

→ get the address of



Pointer Cheat Sheet

- `*`
 - If used in declaration (which includes function parameters), it creates the pointer
 - Ex: `int *p;` // p will hold an address to where an int is stored
 - If used outside a declaration, it dereferences the pointer
 - Ex: `*p = 3;` // goes to the address stored in p and stores a value
 - Ex: `cout << *p;` // goes to the address stored in p and fetches the value
- `&`
 - If used in a declaration (which includes function parameters), it creates and initializes the reference
 - Ex: `void fun(int &p);` // p will refer to an argument that is an int by implicitly using *p (dereference) for p
 - Ex: `int &p = a;` // p will refer to an int, a, by implicitly, using *p for p
 - If used outside a declaration, it means “address of”
 - Ex: `p=&a;` // fetches the address of a (only used as rvalue) and store the address in p

Demo

```
access.engr.orst.edu - PuTTY
1 #include <iostream>
2
3 using namespace std;
4
5 void swap(int a, int b) {
6     cout << "Values at START of swap(): a = " << a << " b = " << b <<
    endl;
7     int temp = a;
8     a = b;
9     b = temp;
10    cout << "Values at END of swap(): a = " << a << " b = " << b << e
    ndl;
11
12 }
13
14
15 int main() {
16
17     int a = 1, b = 2;
18     cout << "Before swap: a = " << a << " b = " << b << endl;
19     swap(a,b);
20     cout << "After swap: a = " << a << " b = " << b << endl;
21
22     return 0;

```

22,1-8 Top

Type here to search 3:55 PM 2/14/2018

```
1 #include <iostream>
2
3 using namespace std;
4
5 void swap(int& a, int& b) {
6     cout << "Values at START of swap(): a = " << a << " b = " << b <<
    endl;
7     int temp = a;
8     a = b;
9     b = temp;
10    cout << "Values at END of swap(): a = " << a << " b = " << b << e
    ndl;
11
12 }
13
14
15 int main() {
16
17     int a = 1, b = 2;
18     cout << "Before swap: a = " << a << " b = " << b << endl;
19     swap(a,b);
20     cout << "After swap: a = " << a << " b = " << b << endl;
21
22     return 0;
```

"pass_by.cpp" 23L, 433C

5,1

Top

```
1 #include <iostream>
2
3 using namespace std;
4
5 void swap(int* a, int* b) {
6     cout << "Values at START of swap(): a = " << a << " b = " << b <<
    endl;
7     int temp = *a;
8     *a = *b;
9     *b = temp;
10    cout << "Values at END of swap(): a = " << a << " b = " << b << e
    ndl;
11
12 }
13
14
15 int main() {
16
17     int a = 1, b = 2;
18     cout << "Before swap: a = " << a << " b = " << b << endl;
19     swap(&a, &b);
20     cout << "After swap: a = " << a << " b = " << b << endl;
21
22     return 0;
```

"pass_by.cpp" 23L, 439C

19,2-9

Top


```
flip3 ~/teaching/cs161/lectures/week_6 160% a.out
Before swap: a = 1 b = 2
Values at START of swap(): a = 0x7ffcd15ff40c b = 0x7ffcd15ff408
Values at END of swap(): a = 0x7ffcd15ff40c b = 0x7ffcd15ff408
After swap: a = 2 b = 1
flip3 ~/teaching/cs161/lectures/week_6 161% vim pass_by.cpp
flip3 ~/teaching/cs161/lectures/week_6 162% █
```